Network Analytics for 5G Data

Rui Pedro Passos Ferreira

Relatório de Estágio para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Dr. Hugo Proença
Co-orientador: Dr. Eng. Marco Araújo

Outubro de 2023

Declaração de Integridade

Eu, Rui Pedro Passos Ferreira, que abaixo assino, estudante com o número de inscrição M11911 do Mestrado em Engenharia Informática da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 09/10/2023

Dedication

To my family and friends.

Acknowledgements

Firstly, I would like to thank my supervisor, Prof. Dr. Hugo Proença, for his support, guidance and insights throughout this research project. His expertise in the domain of Artificial Intelligence and his insightful recommendations significantly contributed to the accomplishment of this research endeavor.

I would like to extend my sincere appreciation to Eng. Dr. Marco Araújo for presenting me with a challenging opportunity. His continuous trust and confidence in my abilities acted as a strong motivation to push the limits of my research and pursue its potential.

Additionally, I am thankful to the R&D team at Capgemini Engineering, specifically Adriano Goes, Afonso Teixeira, Bruno Mendes, João Donato, João Fonseca and Raul Barbosa. Their contributions in the form of valuable discussions and insights were instrumental during the project's development. Their input played a significant role in shaping my ideas, refining the methodology employed and enhancing the overall quality of my work.

I would also like to acknowledge COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI) and the European Regional Development Fund (ERDF) for their indispensable co-funding support for the 5G/SDN Intelligent Systems For LOw latencY V2X communications in cross-Domain mobility applications (FLOYD) project (POCI/01/0247/FEDER/047222).

Lastly, I could not have undertaken this challenge without the continuous encouragement of my family. Their belief in my capabilities has been my guiding light, helping me overcome obstacles and maintain a positive outlook during trying times. I am genuinely grateful for their love and assistance at every step of this demanding journey.

Resumo

Durante a última década, os avanços na infraestrutura da rede e inteligência artificial impulsionaram o desenvolvimento e integração dos veículos autónomos na sociedade.

As redes 5G permitem através da latência ultra-baixa e a alta largura de banda integrada, uma permuta de dados instantânea entre veículos, infraestruturas e sistemas de inteligência artificial de computação na nuvem. Esta conetividade potencia o desenvolvimento de sistemas de transporte inteligentes. Os algoritmos de aprendizagem automática a bordo destes veículos e integrados nas infraestruturas de rede permitem a análise de dados em tempo real, garantindo um processo de decisão mais preciso e uma maior segurança e sustentabilidade rodoviária.

Neste âmbito, esta investigação avalia várias abordagens com vista ao desenvolvimento de sistemas inteligentes com a capacidade de prever com eficácia os valores da latência da rede, a fim de garantir uma comunicação segura em cenários de *platooning*. Os atrasos nos padrões de tráfego são igualmente analisados de modo a extrair conhecimentos relevantes que fomentem estratégias eficientes de gestão do tráfego.

O trabalho descrito neste documento está integrado no projeto europeu 5G/SDN Intelligent Systems For LOw latencY V2X communications in cross-Domain mobility applications (FLOYD), que visa o desenvolvimento de tecnologias para suportar as comunicações exigidas pela condução autónoma, nomeadamente para *platooning*.

Palavras-chave

5G, Aprendizagem Automática, Análise de Dados, FLOYD, Inteligência Artificial, Platooning

Resumo Alargado

Durante a última década, os avanços na infraestrutura da rede e inteligência artificial impulsionaram o desenvolvimento e integração dos veículos autónomos na sociedade.

As redes 5G permitem através da latência ultra-baixa e a alta largura de banda integrada, uma permuta de dados instantânea entre veículos, infraestruturas e sistemas de inteligência artificial de computação na nuvem. Esta conetividade potencia o desenvolvimento de sistemas de transporte inteligentes. Os algoritmos de aprendizagem automática a bordo destes veículos e integrados nas infraestruturas de rede permitem a análise de dados em tempo real, garantindo um processo de decisão mais preciso e uma maior segurança e sustentabilidade rodoviária.

Neste âmbito, a investigação descrita no presente documento tem por objectivo avaliar diversas abordagens com vista ao desenvolvimento de sistemas inteligentes com a capacidade de prever eficazmente os valores de latência da rede. Este atributo de rede é um elemento ímpar para garantir uma comunicação segura sem interrupções em cenários de *platooning*. Este cenário remete a um grupo de veículos que viaja de forma conjunta e sincronizada, que opera de acordo com as indicações enviadas por um veículo líder.

Desta forma, garantir baixas latências é essencial para assegurar a comunicação instantânea entre os veículos e as infraestruturas, a fim de responder a diversos fatores do ambiente que possam interferir no seu percurso e garantir o seu correto funcionamento. Para este fim, recorreu-se ao uso de algoritmos convencionais de aprendizado automática e à implementação de modelos de *deep learning*, nomeadamente uma *Long Short-Term Memory*.

Os contributos deste trabalho estão ainda associados a avanços na otimização dos recursos de rede, através da validação de uma proposta de arquitetura de orquestração que visa monitorizar dados da *Radio Access Network* e otimizar dinamicamente a alocação de recursos na rede por meio de análises de dados em tempo real. Na arquitetura proposta são introduzidas novas abordagens comparativamente às que são definidas pelo 3rd Generation Partnership Project (3GPP) no âmbito da análise de dados da rede.

O trabalho detalhado neste documento está integrado no projeto europeu 5G/SDN Intelligent Systems For LOw latencY V2X communications in cross-Domain mobility applications (FLOYD), que visa o desenvolvimento, validação e exploração de um conjunto de componentes e tecnologias que permitem um desenvolvimento de uma estrutura segura de suporte às comunicações requiridas pela condução autónoma, nomeadamente para *platooning*.

Abstract

Over the last decade, advances in network infrastructure and Artificial Intelligence have driven the development and integration of autonomous vehicles into society.

The ultra-low latency and high bandwidth of Fifth Generation Mobile Networks (5G) enable real-time data exchange among vehicles, infrastructure and cloud-based Artificial Intelligence systems. This connectivity empowers the development of Intelligent Transportation Systems. Machine Learning algorithms onboard these vehicles and integrated into network infrastructures enable real-time data analysis, allowing for a more accurate decision-making process and improved road safety and sustainability.

Within this scope, this research assesses various approaches to building intelligent systems capable of effectively predicting End-to-End latency values to ensure reliable communication in vehicular platooning scenarios. Traffic pattern delays are also analyzed to extract meaningful insights that promote efficient traffic management strategies.

The work described in this document is part of the European project 5G/SDN Intelligent Systems For LOw latencY V2X communications in cross-Domain mobility applications (FLOYD), which aims to advance autonomous driving technologies for Vehicle-to-Everything (V2X) platooning applications.

Keywords

5G, Artificial Intelligence, Data Analysis, FLOYD, Machine Learning, Vehicular Communications, Platooning

Contents

List of Figures

List of Tables

List of Acronyms

3GPP 3rd Generation Partnership Project

5G Fifth Generation Mobile Networks

5GC 5G Core

5G-NR 5G New-Radio

AI Artificial Intelligence

AMF Access and Mobility Management Function

ANN Artificial Neural Network

AUSF Authentication Server Function

ARIMA Autoregressive Integrated Moving Average

B5G Beyond 5G

BBU Baseband Unit

BL Bayesian Learning

BP Back Propagation

C-ITS Cooperative Intelligent Transportation Systems

CSP Communication Service Provider

CMU Carnegie Mellon Portugal

CN Core Network

CNN Convolutional Neural Network

CP Control Plane

CPRI Common Public Radio Interface

CU Central Unit

CUPS Control and User Plane Separation

DECOR Dedicated Core Networks

DL Downlink

DN Data Network

DNN Deep Neural Network

DU Distributed Unit

E2E End-to-End

EDA Exploratory Data Analysis

EFB Exclusive Feature Bundling

eMBB Enhanced Mobile BroadBand

EPC Evolved Packet Core

ERDF European Regional Development Fund

FCT Foundation for Science and Technology

FFNN FeedForward Neural Network

FLOYD 5G/SDN Intelligent Systems For LOw latencY V2X communications in cross-Domain mobility applications

GBDT Gradient Boosting Decision Tree

GCN Graph Convolutional Network

GML Machine Learning on Graphs

GMM Gaussian Mixture Model

gNB Next Generation NodeB

GOSS Gradient-based One-Side Sampling

GRU Gated Recurrent Unit

IQR Interquartile Range

ISP Internet Service Provider

IT Instituto Telecomunicações

ITS Intelligent Transportation System

ITU International Telecommunication Union

IoT Internet of Things

KDE Kernel Density Estimation

KNN K-Nearest Neighbors

KPI Key Performance Indicator

LoA Level of Automation

LGBMR Light Gradient Boosting Machine Regressor

LOCF Last Observation Carried Forward

LR Linear Regression

LSTM Long Short-Term Memory

LTE Long-Term Evolution

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

MEC Multi-Access Edge Computing

MIMO Multiple-Input Multiple-Output

ML Machine Learning

MLP Multi-Layer Perceptron

mMTC Massive Machine Type Communications

mmWave Millimeter wave

MSE Mean Square Error

NEF Network Exposure Function

NF Network Function

NFV Network Function Virtualization

NG-RAN Next-Generation RAN

NMS Network Management System

NOCB Next Observation Carried Backward

NRF Network Repository Function

NS Network Slicing

NSA Non-Standalone Architecture

PCAP Packet Capture

PCF Policy Control Function

PDF Probability Density Function

PLR Packet Loss Rate

PON Passive Optical Network

QoS Quality of Service

QUIC Quick UDP Internet Connections

R^2 Coefficient of Determination

RAN Radio Access Network

RAT Radio Access Technology

RBF Radial Basis Function

RF Random Forest

RMSE Root Mean Square Error

RMSRE Root Mean Square Relative error

RNN Recurrent Neural Network

RPE Relative Prediction Error

RPi Raspberry Pi

RRU Remote Radio Unit

RSU Road Side Unit

RSRP Reference Signal Received Power

RSRQ Reference Signal Received Quality

RSSI Received Signal Strength Indicator

RTT Round-Trip Time

SA Standalone Architecture

SAE Society of Automotive Engineers

SARIMA Seasonal Autoregressive Moving Average

SBA Service-Based Architecture

SBI Service-Based Interface

SCP Service Communication Proxy

SDN Software-Defined Networking

SGD Stochastic Gradient Descent

SLSTM Smoothed LSTM

SMF Session Management Function

SoA State-of-the-Art

SS Signal Strength

STHGCN Spatio-Temporal Hybrid Graph Convolutional Network

SVR Support Vector Regressor

SVM Support Vector Machine

TPE Tree-Structured Parzen Estimator

TS Technical Specification

UC Use Case

UE User Equipment

UP User Plane

UPF User Plane Function

uRLLC Ultra-Reliable and Low Latency Communications

V2I Vehicle-to-Infrastructure

V2N Vehicle-to-Network

V2P Vehicle-to-Pedestrian

V2V Vehicle-to-Vehicle

V2X Vehicle-to-Everything

VGG Visual Geometry Group

WoWMoM World of Wireless, Mobile and Multimedia Networks

XGBoost Extreme Gradient Boosting

Chapter 1

Introduction

This chapter presents the motivating factors that prompted the research conducted in this document, outlines the proposed objectives for the investigation, and highlights the potential contributions it could make in various domains. Furthermore, it provides an overview of the document's structure. Finally, a brief description of the European project linked to the work described in this document is also included.

1.1 Context and Motivation

In the coming years, autonomous and connected vehicles will become omnipresent in society, transforming transportation and redefining how people commute and travel.

Autonomous driving refers to self-driving vehicles or transport systems that operate without human intervention. These vehicles perceive their surroundings through integrated cameras and sensors, enabling them to make intelligent decisions based on this information. Achieving this level of automation implicates developing and seamlessly integrating advanced technologies and algorithms across various domains.

One of the primary factors driving the growth of autonomous vehicles is the introduction of new technologies in 5G networks. However, it is important to note that this dissertation will not cover all aspects of 5G technology applied to autonomous driving. Instead, the focus will be on analyzing its specific applicability to platoons within autonomous driving.

Harnessing the data provided by communication infrastructure and remote computing capabilities enhances vehicle dynamics within Intelligent Transportation System (ITS). However, the seamless delivery of connectivity and computing services to vehicles poses a significant challenge. This challenge is rooted in ensuring uninterrupted connectivity and reliable computing capabilities in various environments, including urban areas with high network congestion and remote locations with limited infrastructure. Additionally, addressing the security and privacy problems associated with transmitting and storing sensitive data within vehicles is essential to fully realizing these advancements' potential.

The fusion of Artificial Intelligence (AI) with groundbreaking advancements in edge computing and 5G resulted in the creation of a complete set of tools for networking and communications to support platooning applications. Platooning refers to a group of vehicles traveling together. All the platoon vehicles receive frequent data from the leading vehicle to manage their operations in sync [1].

These tools enable seamless communication and coordination between vehicles in a platoon, allowing them to operate synchronized. Therefore, this transformation contributes to the sustainability of the transportation ecosystem. It also promotes efficiency and reduces traffic congestion, ultimately improving the overall quality of transportation [2].

The research conducted in this document is integrated in the scope of the FLOYD Project [3], a Carnegie Mellon Portugal (CMU) Portugal Large Scale Collaborative initiative, that focuses on developing and implementing cutting-edge solutions to enhance transportation systems' efficiency, safety and sustainability.

The FLOYD project aims to revolutionize Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication by harnessing the capabilities of edge computing and 5G technology. These abilities promote real-time data analysis and decision-making to enhance traffic management, optimize congestion and ensure seamless communication to coordinate all components between platoons and network infrastructure.

Furthermore, this investigation, accomplished at Capgemini Engineering, not only aligns with the project's objectives but also aims to deliver valuable insights for the future advancement and acceleration of the integration of platooning applications into society.

1.2 Objectives and Contributions

The mobility industry is experiencing a significant increase in the integration of autonomous features towards fully autonomous vehicles' development.

Advancements in AI and advanced networking are facilitating the rapid integration of autonomous vehicles into society by aiding in developing dedicated infrastructure and implementing orchestration systems. Closed-loop orchestration plays a pivotal role in this process by enabling seamless coordination and communication among various components of the autonomous vehicle ecosystem, including sensors, actuators and decision-making algorithms [4].

This dissertation was developed within the context of an enterprise environment and is part of a European Project with a partnership involving CMU in the USA, known as the FLOYD project. This project is dedicated to designing a comprehensive network infrastructure to support V2X applications, specifically on vehicle platooning, which involves a group of vehicles traveling closely together in a coordinated manner.

As depicted in Figure 1.1, FLOYD involves a complex architectural system that requires the integration of various components and technologies. Cooperatively, these elements create a robust infrastructure to support V2X communications, especially in the context of platooning.

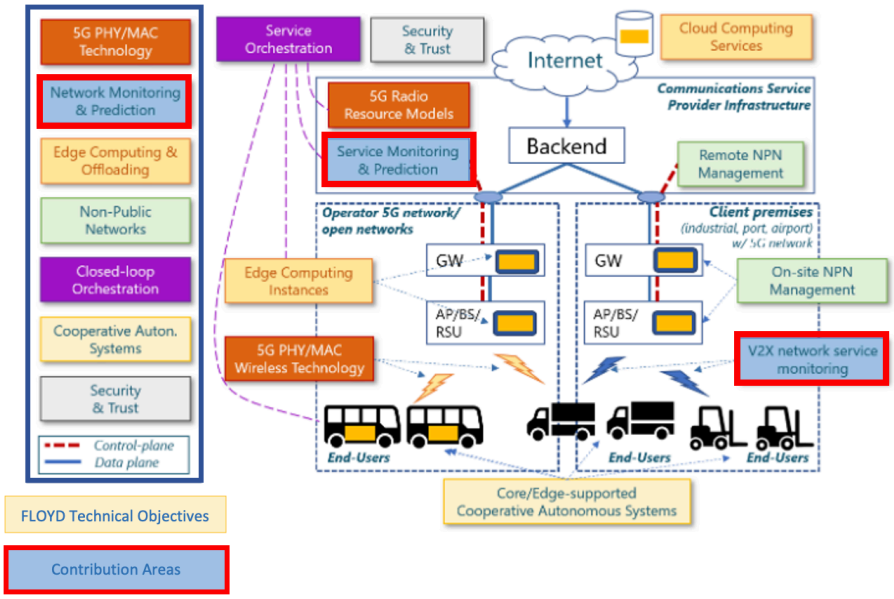


Figure 1.1: FLOYD Project - High Level Overview

As part of the technical objectives defined for the project, this research is integrated into Network Monitoring and Prediction, which incorporates various techniques and methodologies for monitoring and predicting network performance, identifying potential issues, and optimizing the overall V2X communication system.

Within the domain of network monitoring and prediction, the primary objective of this study is to investigate the application of AI techniques to develop multiple approaches for constructing an intelligent monitoring system that can accurately predict network service performance for cooperative vehicles, explicitly focusing on the E2E network latency.

In the scope of autonomous platoons, network latency plays a crucial role among the defined Key Performance Indicators (KPIs). Low-latency communication is critical for successfully implementing and safely operating vehicular platooning systems. Moreover, it significantly contributes to the overall efficiency, stability, adaptability and effectiveness of platooning, establishing this KPI as a critical factor in realizing the full potential of this technology and, consequently, improving ITS.

Additionally, this research also aims to explore regular patterns and identify potential correlations with daily and routine events in network traffic data. With support from the user's mobility pattern information, this investigation seeks to provide recommendations to decision engines. Subsequently, the orchestrator will enforce these recommendations through resource management and allocation, emphasizing its crucial role in achieving closed-loop orchestration.

The project work benefits from the collaborative efforts of various R&D entities, all dedicated to advancing network monitoring technologies within cooperative vehicle applications, ensuring their secure and reliable integration into society.

1.3 Project FLOYD

This section provides a comprehensive summary of the project's scope and the entities involved in its development. It also describes the contributions of the investigation outlined in this document to the project's objectives.

1.3.1 Project Description

The progressive advancements in AI and networking infrastructure will make autonomous vehicles a pervasive reality in the upcoming years. Autonomous driving can significantly impact society with many benefits, including decreased accidents caused by human error, reduced costs and a positive environmental impact on transportation. ML algorithms can also enhance traffic management systems by optimizing routes and reducing road congestion. However, delivering seamless connectivity and computing services for vehicles is challenging since it requires coordinating and integrating various technologies and infrastructure components.

Simultaneously, the advent of 5G technology holds the potential to revolutionize the automotive industry by facilitating faster and more reliable V2I communication. With its low latency and high bandwidth capabilities, 5G can support real-time data exchange, enabling vehicles to communicate more effectively with each other and with traffic management systems. Additionally, 5G introduces new radio technologies and a redesigned core-network architecture that promotes modularity and flexibility, critical elements for achieving low latency, high reliability, quick reconfiguration and failure recovery. Moreover, this technology harnesses concepts such as Network Function Virtualization (NFV), edge computing and AI to maximize its capabilities.

The FLOYD (POCI-01-0247-FEDER-045912) is co-funded by the ERDF through the Operational Program for Competitiveness and Internationalization (COMPETE 2020) and the Portuguese Foundation for Science and Technology (FCT) via the CMU Portugal Program.

This project seeks to improve mobility services and accelerate the adoption of autonomous vehicles through advances in the networking and computing technologies used to develop complex mobility and autonomy services and functions. It relies on collaboration between industrial and academic R&D entities to achieve this goal, as illustrated in Table 1.1.

Organization Name	Type of Organization	Non-Business Organization	Public/Private
Capgemini Engineering	Company	No	Private
ISEP	Polytechnic Institute	Yes	Public
Altice Labs, S.A.	Company	No	Private
Instituto Telecomunicações	R&D Center	Yes	Private
VORTEX Colab	Collaborative Laboratory	Yes	Private
USA Partnership: Carnegie Mellon University (CMU)			

Table 1.1: List of Participants in the FLOYD project.

1.3.2 Objectives and Contributions

Project FLOYD leverages advanced cellular networks and computational technologies to develop a complete computation-networking solution that supports autonomous vehicles and ensures reliable connectivity. It focuses on a real-world application for cooperative autonomous driving systems: vehicular platooning.

Platooning represents the future of ITS, where a group of vehicles moves together following the guidance of a platoon leader. Its purpose is to improve road safety, reduce fuel consumption and maximize road utilization efficiency, ultimately contributing to a more sustainable and efficient ITS [5]. However, its full integration requires a robust and reliable network infrastructure to support real-time communication and coordination between vehicles in the platoon.

One pivotal element required for the full integration of platooning on the roads is lower latency, as specified by the KPIs defined in the standards for connected vehicles. Latency significantly impacts platooning vehicles' real-time coordination and response, making it critical for maintaining safe and efficient operations.

Ensuring low latency enables seamless communication and rapid data transmission between vehicles, allowing platooning vehicles to react promptly to changes in traffic conditions, such as sudden braking or lane merging, thereby reducing the risk of accidents and improving overall traffic flow [6].

In pursuit of these objectives, the work described in this document contributes to the overall project by exploring various methodologies aimed at predicting network latency delays to identify potential bottlenecks that could impact communication between platoon vehicles in the future. This work benefits from collaboration with other project entities, specifically Instituto Telecomunicações (IT) and Altice Labs, which built the infrastructure from which the measurements were collected for subsequent data analysis.

Furthermore, the work developed also contributes to the development of efficient strategies for managing and optimizing network resources to enhance overall network performance. This was achieved by proposing an E2E 5G orchestration architecture that enables dynamic resource allocation and real-time network resource analysis [7].

Lastly, the project has associated a sequence of Use Cases (UCs) to be executed throughout its development. These UCs are situated at various levels of the network and computing stack, involving different subsets of components within a 5G network with edge computing to support Cooperative Intelligent Transportation Systems (C-ITS).

1.4 Overview of the structure of the document

This section summarizes the structure of the chapters that will guide the document.

Chapter 1 starts by describing the context and motivation behind the purpose of this work. It also outlines the expected objectives and introduces the FLOYD project, a European initiative in which this research is conducted, in Sections 1.2 and 1.3, respectively.

In Chapter 2, an introductory explanation of the fundamental concepts within the study's domain is provided, with a specific focus on 5G and ML. This chapter also includes a literature review in Section 2.3, which aims to comprehend existing research related to the purpose of this study and facilitate a comparison between them. This process also enables identifying and exploring gaps or areas that require further investigation.

Chapter 3 provides a detailed description of each pipeline within the presented data analysis framework, supplying a step-by-step explanation of the methodology followed.

Moreover, Chapter 4 presents the results obtained from the proposed framework, which are thoroughly analyzed and discussed in subsequent sections.

Lastly, Chapter 5 presents the study's conclusions in Section 5.1 derived from the research findings, while Section 5.2 explores potential future research directions that can build upon the current study and address identified limitations.

Chapter 2

Fundamental Concepts and State-of-the-Art

This chapter aims to provide a comprehensive overview of the fundamental components of the 5G technology and an introduction to the most important principles of ML. Moreover, it discusses and compares articles with methodologies and purposes related to the objectives delineated.

2.1 5G Basic Concepts

The 1G network represents the beginning of cellular networks, marking a crucial milestone in the history of telecommunications. Despite its capacity and coverage limitations, it relied on analog technology for voice communication transmission. Subsequent generations, from 2G to 4G, have experienced significant advancements in mobile network technology. These newer generations introduced digital technology, allowing for faster data transmission, improved call quality and the launch of mobile internet capabilities [8].

2G networks introduced the capability to send text messages and provided basic internet connectivity. 3G networks brought even faster internet speeds, enabling multimedia applications such as video streaming and calling. Finally, 4G networks initiate the transformation of mobile communication with even higher internet speeds, facilitating the implementation of advanced services and the growth of Internet of Things (IoT) [8].

However, the rapidly increasing demand for faster and more reliable connectivity, technological advancements and the emergence of new services underlines the necessity to develop a more robust and dependable network infrastructure.

In this sense, 5G has emerged to transform the communication paradigm, revolutionizing how the world connects and interacts with technology. Compared to the previous generation, it delivers higher data rates ranging from 10 Gbps to 50 Gbps, greater capacity to support more devices simultaneously and lower latency demanded for critical applications. These characteristics make it a pivotal technology in various fields, including autonomous vehicles, remote surgery and intelligent cities. With its ability to handle massive numbers of devices simultaneously, 5G enables seamless connectivity and empowers the IoT to reach its full potential.

One of the critical technologies that enable the unlocking of the full capabilities of 5G is Millimeter wave (mmWave) communications [9].

These communications utilize higher frequency bands, typically above 24 GHz, to transmit data at extremely high speeds.

However, they are associated with the drawback of possessing a limited range and heightened vulnerability to obstacles such as buildings and vegetation. This new generation of mobile networks employs advanced technologies such as beamforming and massive Multiple-Input Multiple-Output (MIMO) to overcome these challenges [10].

Beamforming allows for precise signal transmission, directing signals toward specific devices or areas, effectively compensating for the shorter range of mmWave. On the other hand, massive MIMO uses multiple antennas to transmit and receive data simultaneously, improving efficiency and expanding the overall network coverage. Therefore, deploying a dense network of small cells, in conjunction with these advanced technologies, becomes indispensable to ensure continuous coverage and connectivity in 5G networks.

Advances in hardware components, such as the techniques mentioned, are not the only factors determining the evolution of the 5G network. The most significant advancements are focused on accelerating the software-driven transformation of networks by implementing concepts like Software-Defined Networking (SDN), NFV, Network Slicing (NS) and Multi-Access Edge Computing (MEC) [11].

SDN and NFV facilitate dynamic network management and the virtualization of network functions, offering the flexibility needed to adapt to the diverse requirements of 5G applications. Furthermore, NS divides a physical network into multiple virtual network slices [12]. These slices are designed to have certain capabilities and resources that are customized according to the requirements of a Communication Service Provider (CSP). Lastly, MEC relocates computation and storage closer to end-users, reducing latency and enhancing the 5G experience [13].

2.1.1 5G: Service-Based Architecture

3GPP Technical Specification (TS) Release 15, developed by the 3GPP, an international organization responsible for establishing mobile communications standards, introduces the 5G Core (5GC) architecture and defines the principles and concepts that guide its design. One of these fundamental principles is the concept of Control and User Plane Separation (CUPS), which segregates the control and user plane network functions to enhance network flexibility and scalability [14]. This separation allows for independent scaling and evolution of each plane, thus facilitating more efficient resource allocation and management [15].

Network Functions (NFs) can be deployed as software instances on a virtual infrastructure by decoupling the control and user plane functions. This deployment enhances network agility, simplifying the introduction of new services and technologies without disrupting the entire network infrastructure.

Figure 2.1 presents an overview of the 5GC Service-Based Architecture. These two concepts represent the main difference from the previous generations of mobile communications.

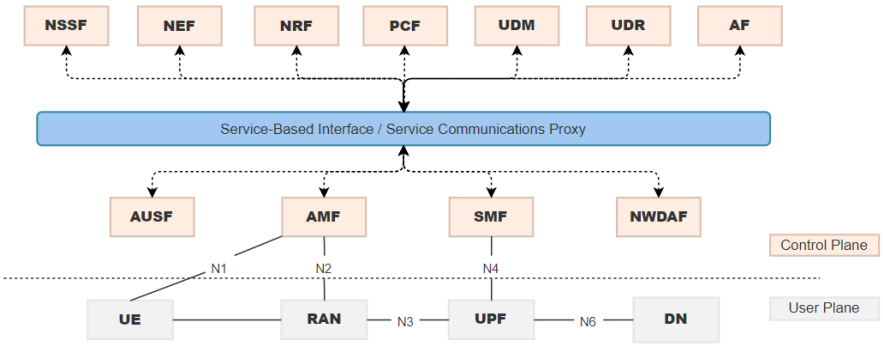


Figure 2.1: 5GC Service-Based Architecture [14]

As illustrated, the Core Network (CN) comprises various NFs, with a separation between Control Plane (CP) and User Plane (UP) functions.

The UP is responsible for managing data traffic between User Equipment (UE) and Data Network (DN). It consists of the UE, the Radio Access Network (RAN), the User Plane Function (UPF) and the DN [14]. The interfaces between the UP functions are defined as reference points (N1–N6 interfaces), as shown in Figure 2.1 [14].

Regarding the CP, it manages the overall operation within the core network and ensures interoperable communication between UE and the network. Furthermore, the CP contains several NFs, including Access and Mobility Management Function (AMF), Session Management Function (SMF), Policy Control Function (PCF) or Authentication Server Function (AUSF). These NFs collaborate to establish and manage connections, allocate network resources and enforce policies for the UE. Their communication is established through a Service-Based Interface (SBI), enabling standardized communication.

The SBI can be implemented using HTTP/2 on top of TLS/TCP as the transport protocol and REST API requests. Additionally, Quick UDP Internet Connections (QUIC), an emerging transport protocol, opens the door for HTTP/3 to support future SBI communications [14].

Each CP NF acts as a service consumer or a service producer, depending on its specific role within the network architecture. The Network Repository Function (NRF) supports finding suitable service producers since it maintains a repository of all available NF instances and their associated services [16]. Each NF registers its services with the NRF, facilitating other network components to discover and use them.

Moreover, the Network Exposure Function (NEF) provides an interface for external applications to access network services and resources [16].

This allows external components to harness the capabilities of NFs by operating as an intermediary between external applications and NFs, ensuring secure and controlled access to network resources. In Release 16 [17], 3GPP introduces the Service Communication Proxy (SCP) to unify standard NF processes into a consolidated platform.

There are two available options for establishing interactions between consumers and producers of NFs services. In the request-response model option, an NF service consumer sends a request to a producer's target service and receives a response [1]. The second option relates to the subscribe-notify model, an NF service consumer subscribes to a service for specific events and receives notifications from the producer when subscribed events occur until the subscription is canceled or expires [1].

2.1.1.1 5G Architectures: Non-Standalone and Standalone

As elaborated in the preceding sections, the release of 5G signifies a substantial advancement over preceding iterations of cellular networks, principally attributable to the incorporation of Next-Generation RAN (NG-RAN) and a Service-Based Architecture (SBA) in the core infrastructure.

The combination of the NG-RAN and a SBA empowers 5G networks to operate independently and seamlessly across diverse network infrastructures and technology generations. Unlike previous mobile networks, where the RAN and core were designed together, 5G considers a disaggregated architecture with greater flexibility and scalability. This separation enables operators to upgrade or replace specific components without disrupting the entire network, which improves efficiency and cost-effectiveness.

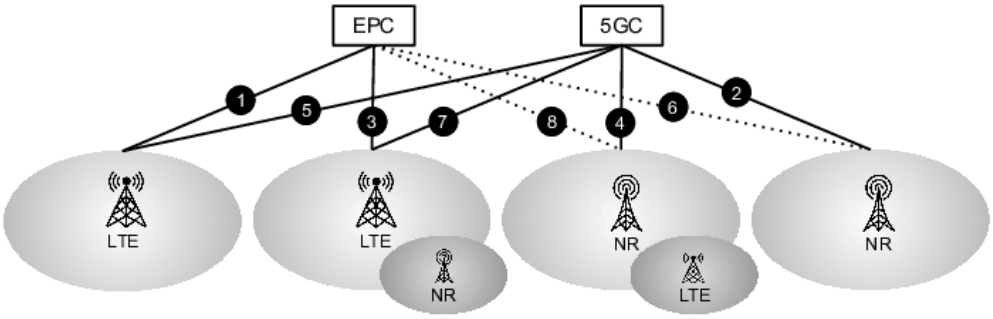
Furthermore, introducing SBA facilitates the deployment of new services and applications that aim to enhance user experiences and promote use cases like autonomous vehicles and remote surgery.

As a result of these advancements, the 5G architecture enables the integration of elements from different generations in various configurations during the transition from 4G to 5G, as demonstrated by the set of available options in Figure 2.2.

In Figure 2.2, Option 1 represents the current 4G system and has been standardized in previous releases of 3GPP. Options 6 and 8 have been discontinued due to compatibility limitations between 5G New-Radio (5G-NR) and the Evolved Packet Core (EPC).

Subsequently, in Release 15, 3GPP introduced the Non-Standalone Architecture (NSA) for Option 3 and the Standalone Architecture (SA) for Option 2 [16].

The NSA promotes a smooth transition from 4G to 5G by leveraging the existing 4G infrastructure. Consequently, it minimizes the need for extensive infrastructure upgrades, enabling existing 4G services to continue operating without interruption while seamlessly and gradually incorporating new 5G services.



Core \ RAN	LTE only	NR only	LTE with NR for data only	NR with LTE for data only
EPC	Option 1 (4G)	Option 6 (disregarded)	Option 3	Option 8 (disregarded)
5GC	Option 5	Option 2	Option 7	Option 4

Figure 2.2: Options defined during the Initial Phase of the 5G System [16]

Its design introduces several advantages, including a substantial network capacity and throughput increase.

Moreover, this architecture delivers greater flexibility in the user plane functions provided by the EPC network. It is achieved by separating the control and user planes in the gateways, along with dual connectivity through Long-Term Evolution (LTE) and 5G-NR.

The implementation of these capabilities was facilitated by the deployment of technologies such as Dedicated Core Networks (DECOR) and enhanced DECOR, CUPS and the use of 5G-NR as a secondary Radio Access Technology (RAT) [16].

Contrary to NSA architecture, the SA deploys a complete E2E 5G network. Consequently, the entire infrastructure, including base stations (gNodeBs) and the CN, is built upon 5G technology, marking a total transition from the traditional EPC to the more advanced SBA.

The integration of cloud-native concepts into the software architecture of a service-based application (SA) allows for enhanced scalability, flexibility, and agility in the deployment and management of network activities. By leveraging containerization and microservices, operators can dynamically scale resources based on demand, reducing costs and promoting faster deployment of new services and features, improving overall network performance. Therefore, it enables SA to deliver advanced features such as network slicing, enabling operators to create multiple virtual networks with customized characteristics to meet the specific requirements of various applications and users [16].

Thus, this architecture will revolutionize mobile communication networks, contributing to unlocking the full potential of 5G networks.

Figure 2.3 presents an overview of both architectures introduced in 3GPP Release 15 [18] and further improved in subsequent releases.

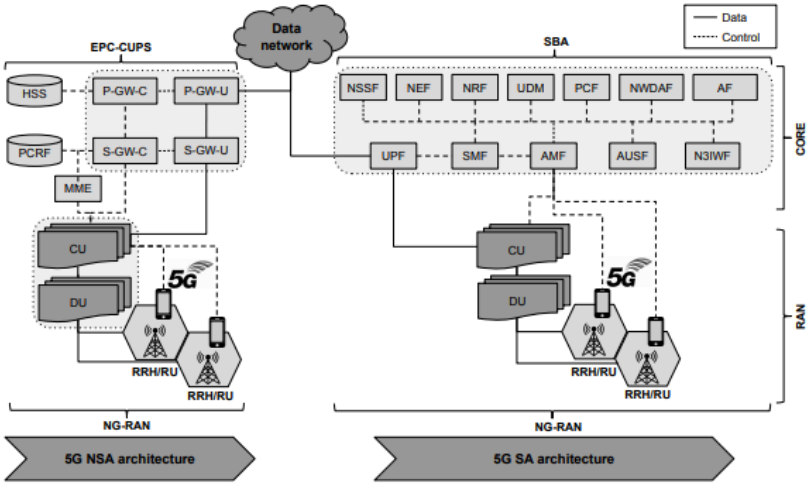


Figure 2.3: NSA and SA [16]

2.1.1.2 Radio Access Network

The RAN is a major component of mobile telecommunications networks, evolving through various technological advancements from 1G and 2G to 3G, 4G and 5G [19]. Each generation of RAN unifies unique technologies and standards to improve network performance, capacity and user experience.

The primary purpose of this network infrastructure segment is to establish radio connections between individual devices and other parts of the network [6]. This is accomplished by utilizing a network of base stations, also referred to as Next Generation NodeBs (gNBs), which transmit and receive radio signals to and from the equipments. A RAN base station consists of several components, including a Baseband Unit (BBU), Remote Radio Unit (RRU), antennas and software-based interfaces.

In traditional RAN configurations, the RRU receives and processes digital radio signals from antennas, transmitting them to the BBU using the Common Public Radio Interface (CPRI), as illustrated in Figure 2.4. The BBU further processes the signal information and forwards it to the core network through backhaul links, which can be either wireless or wired. In reverse, data is transmitted back to the devices. The RAN is also connected to the Network Management System (NMS), allowing it to monitor and manage the signal for correct functionality.

In order to meet the requirements of 5G mobile communications, 3GPP introduced the new RAN architecture known as 5G NG-RAN in Release 15. Subsequent releases, including Release 16 and beyond, have continued to enhance and expand the capabilities of 5G-NR and NG-RAN, addressing various use cases and introducing new features and functionalities. It benefits from advanced technologies like beamforming and massive MIMO to expand coverage and capacity, fostering faster and more reliable user connections.

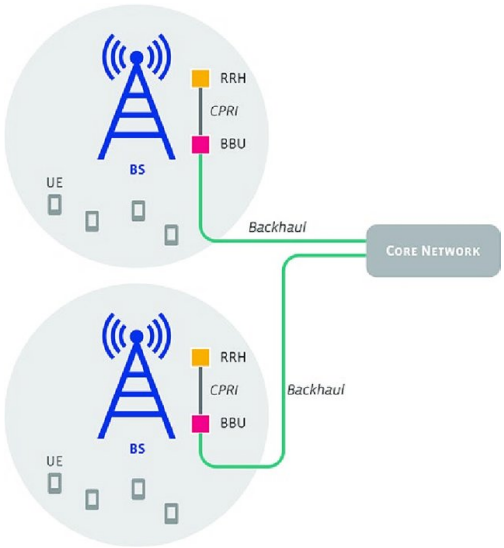


Figure 2.4: Traditional RAN Architecture [20]

Conversely to the 4G RAN architecture, which defined the base station (eNB) as a *monolithic* building block, the 5G NG-RAN architecture adopts a more flexible and modular approach. It enables the deployment of virtualized network functions and separates the base station into distinct functional entities, including the Central Unit (CU) and Distributed Unit (DU).

In this context, the CU operates as the central processing unit within the NG-RAN and is responsible for functions like radio resource management, mobility management and security [21]. It also supports the upper layers of the protocol stack, including SDAP, PDCP, and RRC [22]. On the other hand, the DU functions as a decentralized module situated in close proximity to the antennas. Its primary responsibilities encompass radio frequency processing, antenna management, and beamforming. Both units are interconnected through the NG-RAN, facilitating 5G communication between the UE and the network.

According to the network infrastructure, an NG-RAN node (i.e., a base station) can be configured as a gNB when it operates independently, providing E2E 5G connectivity, or as a ng-eNB when it collaborates with existing 4G infrastructure to deliver 5G services. These configurations enable a gradual evolution from 4G to 5G, providing network operators with deployment flexibility.

The gNBs and ng-eNBs are interconnected through the `Xn interface`, which enables communication and coordination between the different base stations [23]. This interface allows for efficient handover and load balancing between gNBs and ng-eNBs, ensuring smooth connectivity and optimal user performance. They are also connected to the 5GC through `NG interfaces`, specifically to the AMF via the `NG-C interface` and to the UPF via the `NG-U interface` [15].

Figure 2.5 illustrates the relationship between the NG-RAN and the overall 5G infrastructure.

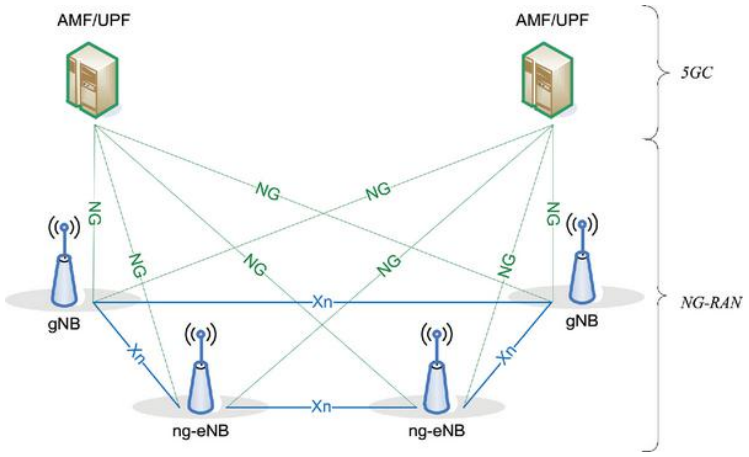


Figure 2.5: Relationship between NG-RAN and 5G System [24])

2.1.2 5G: Use Cases

The emergence of new technologies, as stated in preceding sections, such as NFV, SDN, NS, MEC, mmWave communication and massive MIMO, has revolutionized the mobile telecommunications industry. These technologies deeply impacted network infrastructure and user experiences, increasing network efficiency and enhancing support for various applications and services.

As depicted in Figure 2.6, the International Telecommunication Union (ITU), an organization that collaborates with private and government entities to coordinate global telecommunications networks and services, identified three distinct usage scenarios for 5G communications, as outlined in their vision document [25].

These scenarios are: Enhanced Mobile BroadBand (eMBB), Massive Machine Type Communications (mMTC) and Ultra-Reliable and Low Latency Communications (uRLLC) [27]. Each of them is designed to support to a specific set of service requirements.

The eMBB category represents a direct evolution from 4G LTE networks, aiming to support data-intensive applications that demand higher data rates and more reliable connections, ultimately enhancing user experiences on mobile devices. These requirements are necessary for applications like high-definition video streaming, virtual reality experiences, and online gaming. Additionally, it provides faster download and upload speeds.

In the context of the mMTC scenario, it is intended to accommodate a large number of network devices that have long operational lifetimes and transmit relatively small amounts of non-time-sensitive data. These services are commonly associated with IoT, asset tracking, smart agriculture, smart cities, energy monitoring, smart homes and remote monitoring [28].

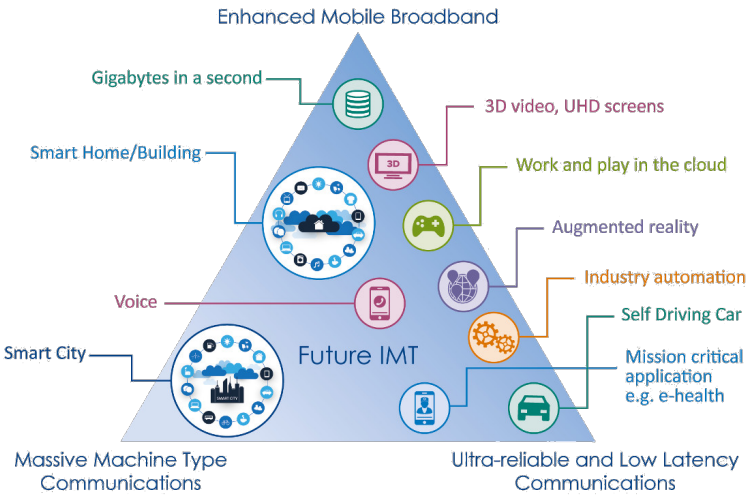


Figure 2.6: 5G main Usage Scenarios defined by ITU [25] [26]

Lastly, the uRLLC scenario focuses on delivering highly reliable and low-latency communication for critical applications. This scenario ensures nearly instant and highly secure data transmission, crucial for use cases like industrial automation, remote surgery and emergency response systems. Additionally, it plays a central role in enabling intelligent transport systems. By harnessing 5G's low latency and safety features, this communication framework helps meet the stringent requirements for supporting V2X communications, which are essential for advancing autonomous driving.

On the other hand, ITU also defines eight KPI to evaluate the performance of 5G networks in the usage scenarios described before [25]. These KPIs encompass latency, data rate, reliability, mobility, connection density, energy efficiency, spectral efficiency and coverage.

Figure 2.7 illustrates the importance of each KPI for each proposed usage scenario, utilizing a three-level scale: high, medium and low importance.

In line with earlier information, eMBB prioritizes user experience data rate, area traffic capacity, peak data rate, mobility, energy efficiency, and spectrum efficiency [30]. However, mobility and user experience data rates may vary in different use cases. For instance, in hotspots, a higher user-experienced data rate is more important than mobility, compared to scenarios with comprehensive area coverage where both factors have different priorities [31].

Regarding the mMTC scenario, a high connection density is required to support numerous devices with low bit rates, mobility and long operational lifetimes.

Within the scope of uRLLC, low latency is the most critical capability to ensure real-time communication and enable instantaneous responses in security-critical applications. At the same time, high data rates may be less critical.

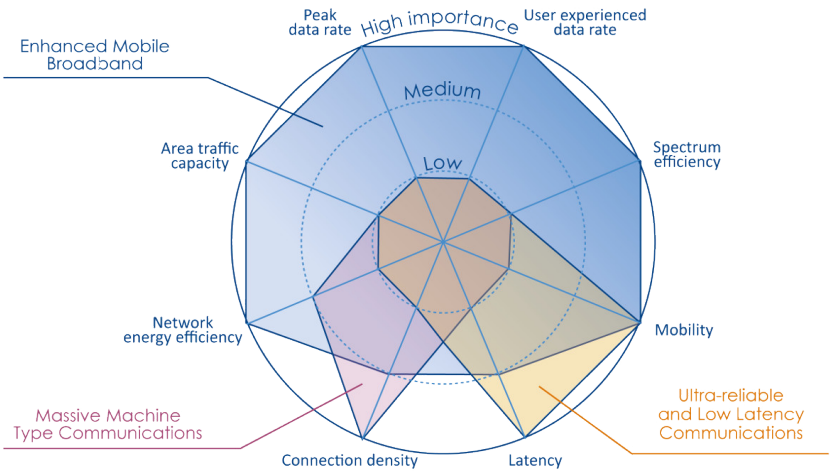


Figure 2.7: Importance of Key Capabilities in different Usage Scenarios [25] [29]

A comparison between the fundamental capabilities of IMT-Advanced (4G LTE) and IMT-2020 (5G) according to ITU-R M.2083 reveals 5G's high performance. This includes achieving peak data rates that are ten times higher than 4G, reducing transmission latency to milliseconds and supporting up to a million concurrent connections per square kilometer, as indicated in Figure 2.8 [25].

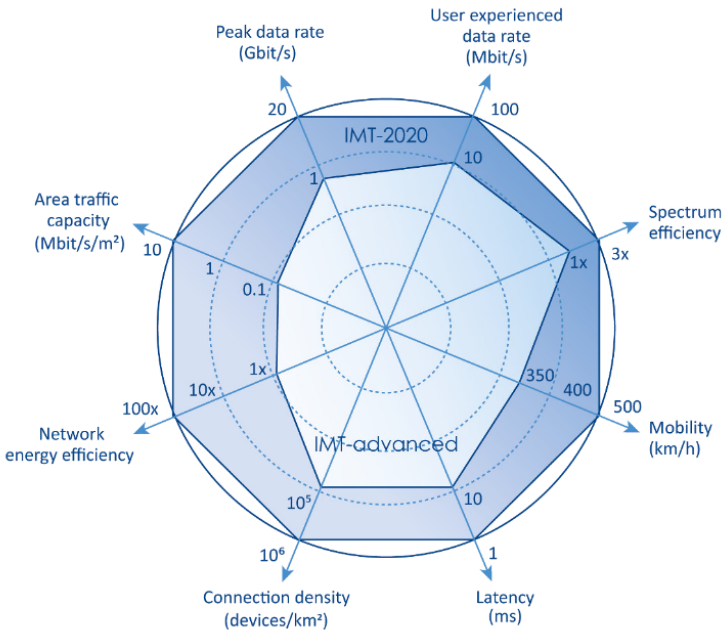


Figure 2.8: Enhancement of Key Capabilities for 5G [25] [29]

2.1.2.1 5G for Autonomous Driving

Within the 5G use scenarios defined in the previous section, uRLLC represents an important milestone in the advancement of V2X applications.

As indicated in [32], V2X applications contain four distinct categories:

- Vehicle-to-Vehicle (V2V)
- Vehicle-to-Infrastructure (V2I)
- Vehicle-to-Network (V2N)
- Vehicle-to-Pedestrian (V2P)

The cooperation of these different types fosters the advancement of a more efficient and secure transportation ecosystem. This progress is achieved through communication and coordination among different elements within the transportation system, enabling real-time information sharing to provide more intelligent services to end users.

Furthermore, the Level of Automation (LoA) is a critical aspect of advanced V2X applications, which influences system performance requirements based on its functional aspects [33]. According to the levels defined by Society of Automotive Engineers (SAE) [34] the LoA includes:

1. No Automation;
2. Driver Assistance;
3. Partial Automation;
4. Conditional Automation;
5. High Automation;
6. Full Automation.

Therefore, various V2X scenarios require transmitting V2X messages, each having unique performance requirements. To address this, 3GPP has established a set of TS covering five key areas: General Aspects (including interworking and communication-related requirements applicable to all V2X scenarios), Vehicle Platooning, Advanced Driving, Extended Sensors and Remote Driving [33].

In this project, the primary focus is on Vehicle Platooning, specifically focusing on reducing network latency. Table 2.1 defines the performance requirements for vehicle platooning across various communication scenarios, as specified by 3GPP.

Communication scenario description		Payload (Bytes)	Tx rate Message	Max E2E latency (ms)	Reliability	Data rate (Mbps)	Min range communication (meters)
Scenario	**Degree**						
Cooperative driving for vehicle platooning Information exchange between a group of UEs supporting V2X application.	Lowest degree of automation	300-400		25	90		
	Low degree of automation	6500		20			350
	Highest degree of automation	50-1200		10	99.99		80
	High degree of automation			20		65	180
Reporting needed for platooning between UEs supporting V2X application and between a UE supporting V2X application and RSU.	N/A	50-1200	2	500			
Information sharing for platooning between UE supporting V2X application and RSU.	Lower degree of automation	6000	50	20			350
	Higher degree of automation			20		50	180

Table 2.1: Performance requirements for Vehicles Platooning defined by 3GPP [33]

2.2 Machine Learning Background

Machine Learning (ML) is a data-driven method in which systems learn iteratively from problem-specific training data to automate the process of developing analytical models and executing associated tasks [35]. It is a subfield of AI that enables computers to discover hidden insights and intricate patterns without explicit programming. Consequently, it plays a crucial role in handling high-dimensional data tasks such as classification, regression, and clustering. In network communication, AI promises to revolutionize the mobile communication industry towards a more intelligent and efficient network [36].

Depending on the problem's objectives and available data, ML can be divided into various subcategories, as shown in Fig 2.9 [37].

Figure 2.9: Types of ML [37]

The following sections will provide detailed overviews of the most commonly used approaches.

2.2.1 Supervised Learning

Supervised learning is a branch of ML that focuses on training models using labeled data. It is considered one of the most commonly used approaches in ML. This approach is characterized by learning a mapping from inputs to output variables and then applying the learned mapping to new unlabeled data [38].

According to different types of output data, supervised learning can be divided into two learning tasks: Classification and Regression [39].

Classification is a supervised learning technique used when the output variable is categorical. There are two classification types: binary classification, where the output variable has two distinct classes, and multi-class classification, where the output variable has more than two different classes.

On the other hand, Regression is another type of supervised learning used in problems where the output variable is continuous. It is divided into two types: Simple Linear Regression, which focuses on the relationship between only two variables, and Multiple Linear Regression, which analyzes the relationship between more than two variables.

2.2.2 Unsupervised Learning

Unlike Supervised Learning, Unsupervised Learning does not require labeled data for model training [40]. Instead, it relies on the inherent structure and patterns within the data to make inferences [35]. This approach is commonly user for clustering, anomaly detection and dimensionality reduction tasks [35].

Clustering refers to grouping elements into clusters based on similarities within the data. It can be particularly useful during the initial data preprocessing phase to detect outliers. The most common technique for this purpose is `K-means clustering`, which involves merging n observations into m clusters by assessing their distances from each cluster's centroid [41].

Additionally, dimensionality reduction aims to decrease the computational complexity of data analysis by reducing the number of variables in the dataset [42]. This technique determines the most important features while eliminating noise and redundancy within the data, ultimately leading to faster and more efficient data processing. It can also be employed as a method to prevent overfitting. There are two primary methodologies employed in the field of dimensionality reduction, namely feature selection and feature extraction [43].

Lastly, the procedure of anomaly detection entails identifying data points, entities, or events that deviate from the data's conventional patterns [44]. This approach is commonly used in various fields, such as finance, cybersecurity and manufacturing.

It has the potential to significantly impact the realm of network communication by aiding in preventing cyberattacks and identifying any anomalies within the network infrastructure that might possibly lead to negative consequences.

2.2.3 Semi-Supervised Learning

Semi-supervised learning is a ML methodology combining a restricted collection of labeled data and a substantial quantity of unlabeled data with training prediction models. Initially, this method constructs a model using the labeled data, which is then refined through iterative integration of the unlabeled data, leading to performance improvements [45].

This approach proves advantageous in scenarios where abundant unlabeled data is readily available or obtaining labeled data is expensive or time-consuming.

2.2.4 Reinforcement Learning

Reinforcement learning implicates single or multiple agents that learn through interactions with their environment. In this process, agents utilize feedback from their actions, presented as rewards or penalties, to assess their behavior to maximize cumulative rewards [46]. Rewards are granted for favorable system decisions, while penalties are imposed for unfavorable ones.

This type of learning is a flexible approach that can be integrated with other ML techniques to solve complex problems, including those related to decision-making, control and optimization [47]. It applies particularly to non-deterministic environments where conditions may change over time or are uncertain.

2.2.5 Deep Learning

Deep learning, a sub-branch of ML, was introduced by [48]. It relies on using Deep Neural Networks (DNNs), which consist of multiple layers of interconnected units, also known as neurons. These networks are employed for automated feature extraction to process and learn tasks directly from input data [49]. This automated feature extraction process forms the core of deep learning, in contrast to conventional ML algorithms, where feature extraction is performed manually [50].

A fully connected DNN consists of an input layer, one or more hidden layers connected sequentially and an output layer that produces outputs. Each layer comprises a specific number of neurons, and the outputs of one layer act as inputs to the subsequent layer. The connections between neurons within each layer are represented by weights. From a mathematical perspective, the architecture of DNNs is differentiable [51].

As a result, the model's weights learn by minimizing a loss function using gradient descent methods and backpropagation, efficiently maximizing the network's performance [51].

There are several different types of neural networks architectures. However, the most popular are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [49].

Convolutional Neural Networks have revolutionized traditional neural networks by automatically learning and extracting features from input data without human supervision [49]. This is achieved through convolutional layers, which apply filters to the input data, enabling the network to identify patterns and spatial relationships. Typically, CNNs are structured with multiple convolutional layers, followed by pooling layers to reduce dimensionality. Lastly, fully connected layers receive the output of the preceding layers and carry out classification or regression tasks.

The use of CNNs has significantly improved performance in various domains, including computer vision tasks such as image recognition, segmentation and object detection [52]. Some of the most commonly used state-of-the-art CNN architectures are: Visual Geometry Group (VGG) [53], AlexNet [54], Inception [55] and ResNet [56].

Recurrent Neural Networks are a popular type of neural network targeted for sequential data structures, such as time-series data and natural language processing [57]. Their architecture considers recurrent connections, which enable the transfer of information from previous steps to the current one, creating memory-like functionality. This capability enables RNNs to preserve context and sequentially identify dependencies among data points over time, unlike conventional DNNs, which presume that inputs and outputs are independent [58].

However, a simple RNN architecture has a limitation of the vanishing gradient problem, which restricts its ability to capture long-term dependencies [49].

In order to address this issue, more advanced variants of RNNs have been developed, such as LSTM [59] and Gated Recurrent Unit (GRU) [60]. These architectures incorporate additional gates that control the flow of information, enabling them to retain or forget information over longer sequences selectively [58]. This enhances their effectiveness in capturing long-term dependencies and mitigates the vanishing gradient problem [61].

2.3 State-of-the-Art

The growing utilization of mobile devices and applications, coupled with the demands introduced by 5G technologies, implies significant changes in mobile and wireless networking infrastructure. 5G and Beyond 5G (B5G) face the challenge of handling high traffic volumes, extracting real-time analytics and agilely managing resources to enhance the quality of services for end-users. Nevertheless, managing complex mobile environments remains a challenging task. The advancements in ML techniques can be pivotal in addressing these challenges.

In this context, articles [51], [39], and [62] present comprehensive surveys related to the integration of ML algorithms to address the challenges posed by 5G.

In the paper [51], the authors focus on deep learning approaches across various domains of mobile networking. Their objective is to present significant research contributions in different areas of cellular networks, compare design principles, guide future research directions and identify unresolved problems worth pursuing with DNNs.

The remaining papers follow a consistent structure, showcasing a variety of successful applications and use cases that harness ML approaches within 5G network systems. They also underscore the need for further investigation into issues within the context of the next generation of network communications.

Furthermore, in alignment with the objectives of this research, the integration of ML algorithms for predicting network attributes, such as E2E network latency, started to earn traction and advancements in earlier generations of mobile telecommunications. The subsequent articles emphasize methodologies relevant to V2X communications.

The study's primary objective [63] is to address the issue of throughput prediction in various scenarios of mobile networks.The authors propose applying a SVR model incorporating historical throughput data and communication quality information, namely the Received Signal Strength Indicator (RSSI). To accomplish this objective, the authors developed an Android application to gather data on throughput and other communication quality parameters at various times of the day and in different scenarios, including different user mobility modes such as stationary, walking, and riding on a bus, using LTE technology. The metrics applied to analyze the model's performance are Relative Prediction Error (RPE) and Root Mean Square Relative error (RMSRE). Lastly, the results demonstrate that using both throughput and RSSI parameters outperforms the SVR model when using only one of them as a parameter.

In [64], the authors again focus on predicting transmission throughput in Wi-Fi networks under varying environmental and network conditions.

The primary distinction of this study from the remaining ones lies in its consideration of multiple applications, including video streaming, video calls, and web browsing, rather than solely focusing on file download applications. To complete this analysis, the authors compiled two datasets: one comprised synthetic data through simulations, while the other consisted of real data via network sniffing. The raw traces obtained from simulations and actual Wi-Fi networks undergo a transformation process in which two data models are employed to extract meaningful and important characteristics: Model I, which computed per-station statistics from the network attributes, and Model II, which computed network-wide attributes and their statistics. The primary distinction between these two models is in the relationship between the number of features and the number of stations. In the first model, as the number of stations increases, the number of features also increases. On the other hand, the second model maintains a constant number of characteristics irrespective of the quantity of stations inside the network.

The study compares data models, revealing that Model-I outperforms Model-II in error performance, generalization, feature extraction and computational efficiency. Besides, the first model also demonstrates faster time and better training efficiency. The evaluation of prediction performance for both data models is afterwards analyzed using four ML approaches. The MLP exhibited superior performance in terms of both prediction accuracy and stability. The subsequent algorithms tested were the Decision Tree, Random Forest and Support Vector Regressor, respectively.

Similar to the methodology in [63], this research [65] involved developing an application to collect real-world data representing communication latency between a vehicle and MEC. This was accomplished by sending ping packets to the IP addresses of both the edge node and the cloud node, respectively. During the preprocessing of the dataset, the authors initially categorized the input data based on signal features such as Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ), and Signal Strength (SS). This categorization aided in classifying the signal and predicting delays. This process resulted in the creating of two datasets: one exhibited a discernible pattern that could be tracked over time, while the other displayed characteristics similar to random noise. Subsequently, the authors employed an LSTM network to capture temporal dependencies in the first dataset. For the second dataset, they proposed a statistical approach that combined the Epanechnikov Kernel [66] and moving average functions. The experimental findings demonstrate that the prediction strategy described in this study effectively decreases the prediction error to 50% of the standard deviation observed in the raw data.

In [67], the authors explored two variations of RNN architectures: Seq2Seq model [68] and LSTM with and without an Attention mechanism [69]. Their objective was to develop a prediction system tailored for time-critical applications with reduced latency. They also used the sliding window method to generate input data for the model.

The results were assessed using RMSE, revealing that the LSTM with an Attention mechanism exhibited the highest performance.

The researchers in [70] developed an approach to evaluate network quality in both LTE and emerging 5G networks, explicitly focusing on throughput in both downlink and uplink scenarios. They collected data from diverse real-life use cases containing various urban, suburban and rural driving conditions and stationary tests in crowded environments. This data included several KPIs of the RAN environment, like RSSI, RSRP and SINR. Due to the limited 5G network data availability, the authors initially trained their models on a conventional LTE network. Subsequently, these models were applied to NSA 5G network data to evaluate their performance and effectiveness. To achieve this, the authors conducted two experiments: one employing the entire dataset and another making predictions on a per-use-case basis. In terms of results, MLP and XGBoost demonstrated the best performance when considering MSE and Coefficient of Determination (R^2) metrics.

In [71], the proposed method applies a two-layer bidirectional LSTM model with input features based on network attributes. The authors inferred that the LSTM model developed is capable of handling dynamic fluctuations in the measurements, and its results outperformed those of SVR, Random Forest (RF), and FeedForward Neural Network (FFNN) models in terms of prediction accuracy. Therefore, a major advantage of this method is its applicability to new regions without the need for extensive measurements beforehand.

In [72], the authors introduce a distinct approach from the remaining ones described. They have formulated a Quality of Service (QoS) prediction as a binary classification problem to determine whether a packet can be delivered within a predefined latency window. Standard ML techniques were employed, like RF and MLP. The Autoregressive Integrated Moving Average (ARIMA) model [73] was also implemented; however, the ML algorithms outperformed it.

The authors of [74] propose a DNN for E2E latency prediction in vehicular communications, considering regression and classification perspectives. The primary differentiation of this research article from other studies is in its novel approach of directly collecting data from the vehicle's integrated communication system while operating under conditions of rapid mobility, enabling remote, realistic and consistent data collection. This represents a significant advancement in data collection for the analysis of vehicle communication. The results indicate that presented DNN-based has superior performance compared to State-of-the-Art (SoA) approaches like RF, LGBMR and LR in continuous latency prediction. Furthermore, their study introduces explainable AI, allowing for inferences about the strong influence of network load on predicted latencies.

In [75], a unidirectional LSTM model was employed to predict traffic patterns in V2X networks under various scenarios, taking into account the rate of transmitted packets per second. This study explored different packet transmission rates per second, specifically 4, 6, 8, 12 and 14 packets/s.

The model's performance was estimated using the metrics RMSE, MAPE and processing time.

The simulation results revealed that experiments involving a packet transmission rate of 4 packets/s exhibited the highest prediction accuracy, surpassing the other rates. Conversely, models operating at a rate of 14 packets/s showed the lowest accuracy in their predictions. Additionally, experiments conducted at a rate of 12 packets/s demonstrated the fastest processing time, whereas those at 14 packets/s experienced the slowest processing time.

The paper [76] investigates the cellular network traffic forecasting problem and identifies its unique spatio-temporal dependency. For this purpose, the authors define a Spatio-Temporal Hybrid Graph Convolutional Network (STHGCN) model combining Graph Convolutional Network (GCN) and GRU to simultaneously capture temporal and spatial dependencies. The model leverages temporal dependency through recent, daily and weekly components, while also leveraging spatial dependency through spatial proximity, functional similarity and recent trend similarity.The authors experimented intending to predict [1, 2, 3, 4]-steps ahead using 48 samples as input. Based on the model's results, it outperformed SoA baselines in real-world telecommunication traffic datasets, demonstrating its effectiveness.

In [77], the research proposed a LSTM architecture for forecasting mobile data traffic. Data were collected over one month from two different eNodeBs to achieve this. In addition to LSTM, the study evaluated the performance of a FFNN and an ARIMA model. The results showed that LSTM outperformed these models. Furthermore, the authors concluded that using more past observations led to more accurate results.

The authors of [78] introduce a novel Smoothed LSTM (SLSTM) architecture designed for 5G traffic prediction. In order to enhance the model's reliability, this architecture adds a data stabilization process before forward propagation and a data sequence restoration process after generating the prediction results. During the experiments, the predictive capabilities of the proposed algorithm are compared to those of conventional approaches, such as LSTM, GRU and CNN. The findings demonstrate that the proposed model significantly enhances the accuracy of 5G traffic prediction. The model's performance is evaluated based on Root Mean Square Error (RMSE), MAE and R^2.

Moving on, the goal of [79] is to provide a thorough analysis towards understanding the potential factors that impact latency prediction within 5G networks. For this effect, it uses KPIs RAN measurements collected at each gNB from various situations such as vehicle mobility, high urban traffic and social gathering events. According to the outcomes, within the features evaluated to estimate the impact on Downlink (DL) latency, the ones that have the most impact are resource utilization in DL, the number of active UEs and the traffic volume in DL, respectively. In this case, the higher their values, the higher the latency.

In addition, the authors do experimental research on probabilistic regression, anomaly detection and predictive forecasting, exploring newly emerging fields within the ML domain, such as Bayesian Learning (BL) and Machine Learning on Graphs (GML).

A LSTM model is implemented in predictive forecasting, incorporating a probabilistic layer as the last layer. The primary objective of this model is to predict the subsequent 15-minute time phase. For training the model, 20 consecutive days were used, and the subsequent ten days for testing. The overall R^2 score reached was 0.75. Lastly, the authors concluded that latency observations in the UP follow a Hypoexponential probability distribution.

In [37], the authors reinforce the importance of traffic flow prediction in the transportation industry and how it can impact the efficiency of transportation systems and improve overall traffic management strategies. In order to fulfill this purpose, an in-depth investigation of the current ML and deep learning methodologies employed in the domain of traffic prediction is provided. Moreover, the challenges presented by AI algorithms in this particular field are inferred.

In order to address the high complexity computational methods for predicting QoS on V2X communication and the absence of environmental features for this type of communication, the paper [80] introduces a novel Informer-based QoS prediction model with a causal convolution self-attention mechanism. The main advantage of this work is the higher performance with minimal computing resources. It focuses on two critical requirements for wireless communication systems: Packet Loss Rate (PLR) and communication delay. Furthermore, the dataset includes data from actual traffic congestion, allowing for a more accurate evaluation of the system's efficacy in complex traffic environments. The experimental results indicate that the proposed model is more accurate and stable than classical prediction methods Back Propagation (BP) neural network [81], ELMAN [82], LSTM [59] and CNN [83].

The article [84] distinguishes itself from the others by analyzing the use of ML algorithms to predict traffic network patterns and maximize energy savings in cellular networks through sleep mode techniques. Initially, the authors conducted an analysis of traffic patterns in cellular networks, demonstrating the daily fluctuations in traffic volume. Then, four ML algorithms and a traditional time series forecasting model were evaluated. Two approaches were applied for evaluation: one considering the complete dataset and another based on individual days of the week.

Among the tested algorithms, RF exhibited the highest performance. Additionally, all ML models outperformed the Seasonal Autoregressive Moving Average (SARIMA) model. Performance evaluation was conducted using the following metrics: Mean Absolute Error (MAE), RMSE, and R^2.

2.3.1 Discussion Literature Review

In summary, one of the significant findings in the literature review is the scarcity of 5G data availability. As stated in previous studies, the majority of the data may not accurately represent the actual scenarios of V2X communication networks, impacting the reliability of ML algorithms when applied to real-world scenarios. This vulnerability highlights the need for additional research and data collection, explicitly targeting 5G networks and V2X communications.

Furthermore, current QoS prediction methods that rely on traditional ML models may not reach the necessary level of accuracy under complex conditions. Conversely, deep learning models have exhibited promising results in enhancing QoS prediction accuracy in complex scenarios. These models can extract and combine abstract features, enabling them to uncover deeper patterns and, consequently, be more suitable for addressing challenges within the dynamic behavior of V2X communication.

Lastly, understanding and predicting network latency in V2X communications is crucial to realizing the maximum potential of connected vehicles and intelligent transportation. It enables the development and refinement of technologies that improve safety, decrease congestion and pave the way for a more sustainable and efficient future in mobility.

Table 2.2 summarizes research papers exploring the AI algorithm's application in predicting networks QoS parameters.

Paper	Goal	Methods and Algorithms	Contributions
[78]	5G Traffic Prediction	SLSTM, LSTM, GRU, CNN	The findings demonstrate that the proposed model significantly enhances the accuracy of 5G traffic prediction
[84]	Analysis of Feature Impact on Latency Predictions	LSTM	The study reveals that the most significant features in DL are resource utilization, the number of active UEs and the traffic volume
[76]	Cellular Network Traffic Prediction	STHGCN	The model demonstrated superior performance compared to SoA baselines in real-world telecommunication traffic datasets
[65]	E2E Latency Prediction	LSTM and statistical approach that combined the Epanechnikov Kernel and moving average functions.	Increase the model's performance through a previous classification of the signal features
[67]	E2E Latency Prediction	Seq2seq and LSTM	Implementation of attention mechanisms in LSTM improves its performance
[71]	E2E Latency Prediction	Bidirectional LSTM	This method offers a significant advantage in its applicability to new regions without the need for extensive measurements beforehand
[72]	E2E Latency Prediction	MLP, RF and ARIMA	A distinct approach from the rest focuses on prediction as a binary classification problem
[74]	E2E Latency Prediction	DNN, RF, LGNMR, LR	The DNN-based model outperforms SoA methods in latency prediction and introduces explainable AI, allowing inferences about network load's significant influence on predicted latencies
[77]	Mobile Traffic Data Prediction	LSTM, ARIMA and FFNN	The study revealed that LSTM outperformed other models and using more past observations resulted in more accurate results
[80]	QoS on V2X communication Prediction	Informer-based QoS prediction model with a causal convolution self-attention mechanism	The main advantage of this work is the higher performance with minimal computing resources
[51], [39], [62] and [37]	AI on Traffic Predictions		These studies delved into the current ML and DL methodologies utilized in traffic prediction
[63]	Throughput Prediction	SVR	Importance of RAN KPIs on predictions
[64]	Throughput Prediction in Wi-Fi networks	MLP, DT, RF and SVR	This study distinguishes itself by examining multiple applications, rather than focusing solely on file download applications
[70]	Throughput Prediction	MLP, XGBoost	Collection of real-life data from diverse use cases and stationary tests in crowded environments to evaluate model performance
[75]	Traffic Patterns in V2X networks Prediction	LSTM	Exploration of different packet transmission rates per second and their impact on traffic pattern prediction
[84]	Traffic Network Patterns Prediction	SVR, CART, KNN, RF and SARIMA	ML techniques have been proven to significantly improve energy efficiency in cellular networks by predicting traffic and implementing sleep modes effectively

Table 2.2: Summary of Research on Network Attributes Prediction

Chapter 3

Intelligent Data Analysis: Proposed Framework

This chapter will describe the proposed Intelligent Data Analysis Framework that drives the data analysis process conducted in this research. This framework aims to develop an intelligent system capable of accurately predicting E2E latency delays in network communication using ML algorithms and statistical analysis techniques to improve network performance and optimize resource allocation. Within the objectives of the FLOYD project, this corresponds to a pivotal task that guides and promotes the acceleration of the integration of platooning applications into society.

Subsequent sections will explore each pipeline stage individually, covering data collection, preprocessing, modeling and evaluation.

3.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) corresponds to initial data exploration to analyze and discover trends and patterns using statistical summaries and graphical representations [85]. This process also ensures that the data are clear and suitable for the ML algorithms processing the data.

The following sections will introduce various techniques to uncover patterns and trends, enabling helpful insight into network behaviours and making informed traffic management and optimization decisions. These techniques include data visualization, statistical analysis and anomaly detection, which collectively contribute to targeted strategies that can be implemented to improve traffic flow efficiency and enhance overall network performance.

3.1.1 Data Collection

Within the scope of this research, the dataset consists of records collected through an extensive measurement of packet latency within the network of a national telecommunications operator. The measures were gathered from the uplink network status, which included various types of network traffic.

The purpose of conducting these measurements was to explore and understand the performance attributes of the network.

Within the KPIs defined in Section 2.1.2.1 for vehicular platooning applications, this work focuses on network latency. Network latency is a data packet's Round-Trip Time (RTT), which refers to the time it takes to travel across the network and back [86]. Various factors can impact latency delays, such as network congestion, distance, processing delays and the quality of the network infrastructure.

Since the FLOYD project is focused on developing advanced communication technologies for V2X applications, it aims to optimize network infrastructure and reduce latency for seamless and reliable data transmission.

Lower network latency ensures that data transmission is efficient and prompt, allowing faster and more responsive network communication between platoon vehicles.

In brief, the symbiotic relationship between V2X communications and low latency is pivotal for achieving the full potential of connected vehicle technologies. Through timely information exchange, these technologies pave the way for safer roads, enhanced traffic flow, cooperative and autonomous driving, pedestrian protection and efficient emergency responses [80].

In this project, a testbed was constructed to gather an extensive dataset on the E2E latency delays. The latency measurements were conducted between a fixed node connected to a 5G NSA commercial network and a remote node connected via a Passive Optical Network (PON), as depicted in Figure 3.1.

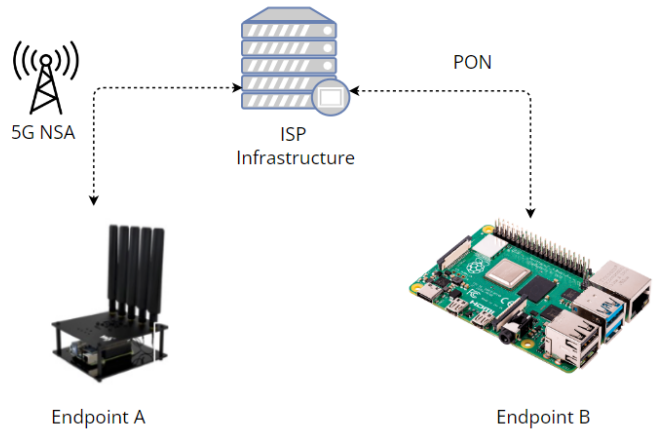


Figure 3.1: Experimental Testbed employed in FLOYD Project

This scenario simulates a 5G node's communication path to an edge server using real-world commercial networks managed by the same Internet Service Provider (ISP). It conducts active network measurements between the endpoints to calculate the corresponding E2E network latency. Packets of varying sizes (100 bytes, 500 bytes, and 1470 bytes) were generated and transmitted across the network at different intervals to collect measurements. Subsequently, the latency samples were recorded in a database for further analysis.

The data collection process was extended by approximately two months, encompassing around 55 days.

The final dataset consists of around 5.000 individual measurements organized into 15-minute intervals, similar to the approach followed in [76] and [79]. This time frame strikes a balance between capturing short-term fluctuations and long-term trends in network E2E latency. Additionally, grouping the data into these intervals makes the dataset more manageable while preserving sufficient information to analyze latency variations.

Each dataset instance includes the timestamp of delay collection and its respective value measured in milliseconds (ms).

3.1.2 Data Preprocessing

Data preprocessing is required inside the project's framework pipeline since it converts raw data into useful information to be further processed by ML algorithms. This process refers to the data's cleaning, transformation and integration procedures [87]. During these steps, any inconsistencies or errors in the data are rectified, missing values are handled, and the data is normalized to improve its quality and suitability for subsequent stages, ultimately leading to more accurate outcomes.

The data analysis presented in this document necessitates some of the data preprocessing procedures mentioned above, which will be elaborated upon in the following sections.

3.1.2.1 Handling Missing Values

A preliminary dataset analysis reveals missing values, as depicted in Figure 3.2. Approximately 5% of the total dataset contains missing values.

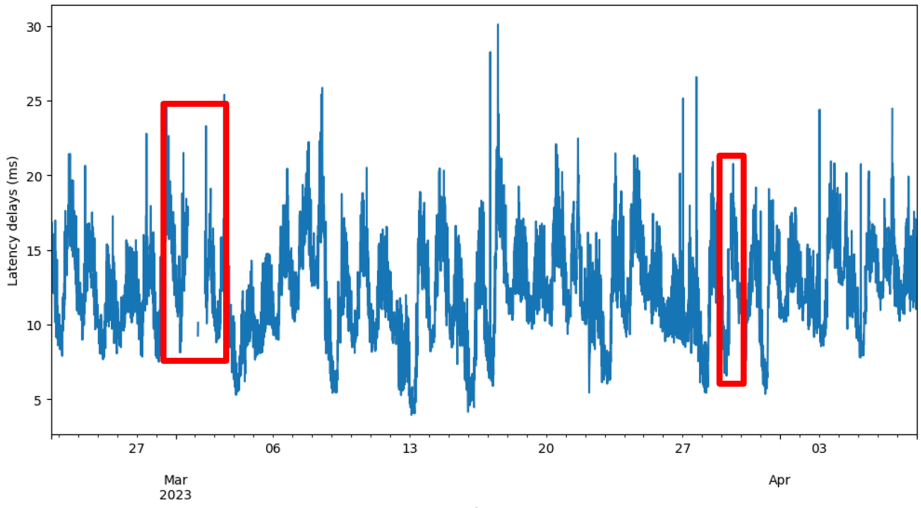


Figure 3.2: Missing Values in the Dataset

These missing values can arise from various factors, such as network failures or data collection errors. Therefore, applying an appropriate solution to handle them is necessary before proceeding with further analysis.

To address this issue, the `Missingno` library [88] was initially imported to investigate the pattern of missingness within the data. This library provides visual representations of missing data, facilitating a quick inspection of the dataset's extent and distribution of missing values.

Figure 3.3 presents the matrix view generated by this library, where black lines indicate non-missing values and white lines represent missing values.

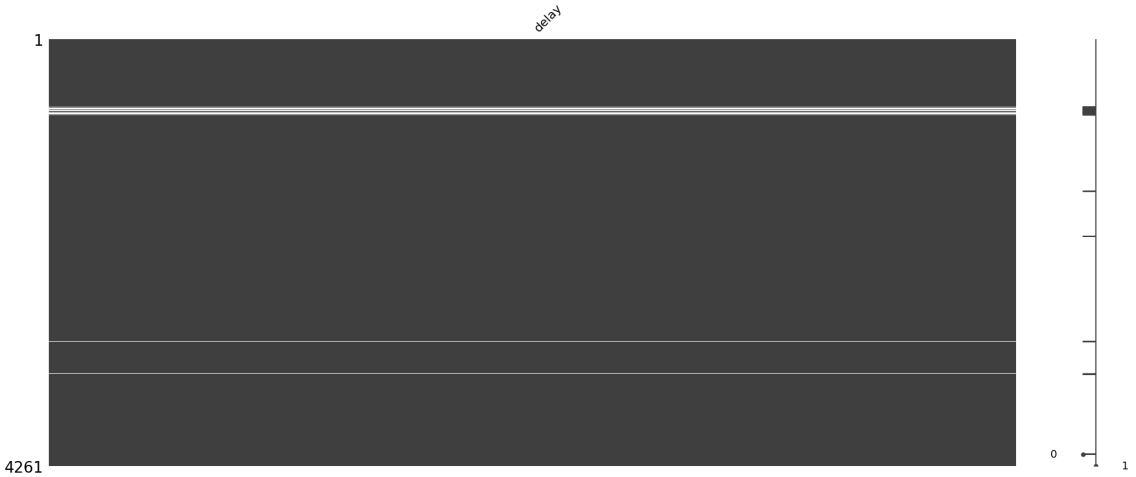


Figure 3.3: Representations of Missing Data by `Missingno` library

The observation suggests that the missingness pattern does not exhibit a specific trend or pattern. As a result, further analyses were conducted to determine the root causes of the missing data.

These analyses revealed the presence of missing values derived from errors in data collection and recording, specifically at one of the endpoints of the testbed. During the collection and recording of measurements, one of the Raspberry Pi (RPi) devices required an upgrade, which led to the loss of route tables for packets and, subsequently, the absence of recorded values for E2E network latency. In another scenario, a device encountered insufficient memory space to store the Packet Capture (PCAP) files.

To address the issue of missing values, several practical strategies can be implemented [89]. However, methods such as Last Observation Carried Forward (LOCF), which replaces null values with the most recent non-missing values, and Next Observation Carried Backward (NOCB), which fills missing values with the following non-missing values, were ruled out. These methods may introduce bias and inaccuracies in the data analysis, mainly when multiple missing values are clustered within specific intervals, as illustrated in Figure 3.2.

Instead, a more robust approach was employed to handle the missing values, namely the `interpolation` technique. This method considers the relationships between variables and provides more reliable estimates for the missing data points. It was implemented using the Python Pandas library through the `interpolate` method [90].

Among the available interpolation methods, two were evaluated: `spline` and `time`. Once the dataset consists of time series data, the most suitable approach is the `time` method. This method uses linear interpolation based on the date and time of the index column, assigning greater weight to nearby data points than distant ones.

Figure 3.4 displays the attained results through this method, where red points symbolize the areas interpolated.

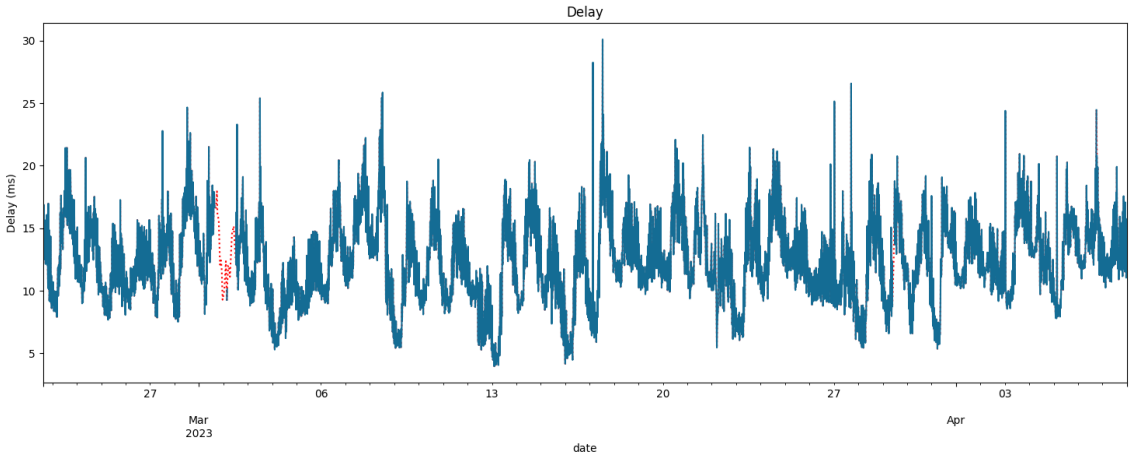


Figure 3.4: Interpolated Areas Using Interpolation Method

In addition to the interpolation method, other advanced techniques were also explored, namely k-Nearest Neighbors (K-Nearest Neighbors (KNN)) [91] and multiple iterative imputations [92]. While the KNN method identifies the k nearest data points to the missing value and uses their values to make an estimation, multiple iterative imputations generate multiple imputed datasets and iteratively refine missing values based on observed data. However, the outcomes produced by both approaches did not match the intended results.

3.1.2.2 Normalization

Normalization is a fundamental data preprocessing procedure in the data analysis pipeline that addresses outliers and promotes uniformity in the distribution of values. It mitigates the impact of outliers by scaling the data points that exhibit extreme values so that they all vary within the same range, minimizing the risk of them influencing the analysis.

Consequently, this procedure enables ML models focus on the more representative underlying patterns and trends in the data, resulting in more accurate predictions and dependable outcomes.

Moreover, normalization improves the comparability and interpretability of results since variables are standardized to a standard scale, facilitating appropriate comparisons and a better understanding of their contributions to the overall analysis.

In this research, the E2E latency values were normalized using the Z-Score technique, as followed in [76].

The standard score (Z-Score) is a statistical method that normalizes values by subtracting the mean from each value and then dividing the result by the standard deviation of the values [93], as stated in equation 3.1:

$$Z = \frac{x - \mu}{\sigma} \quad (3.1)$$

where x represents the observed value, μ is the mean of the sample and σ corresponds to the standard deviation of the sample.

This process transforms the data distribution into a normal distribution with an average of zero and a standard deviation of one.

3.1.3 Data Visualization

Data visualization is the process of graphically representing data to facilitate the recognition and interpretation of patterns, trends, and hidden relationships within the data [94]. Additionally, it aids in identifying outliers in the data. In summary, data visualization promotes a deeper and more intuitive data analysis and decision-making process, ensuring that data-driven decisions are well-informed and result in accurate and meaningful outcomes.

In this context, the initial analysis explores the average daily patterns of network E2E latency delays. Initially, a bar plot was generated to visualize the average delay across different hours of the day, as shown in Figure 3.5.

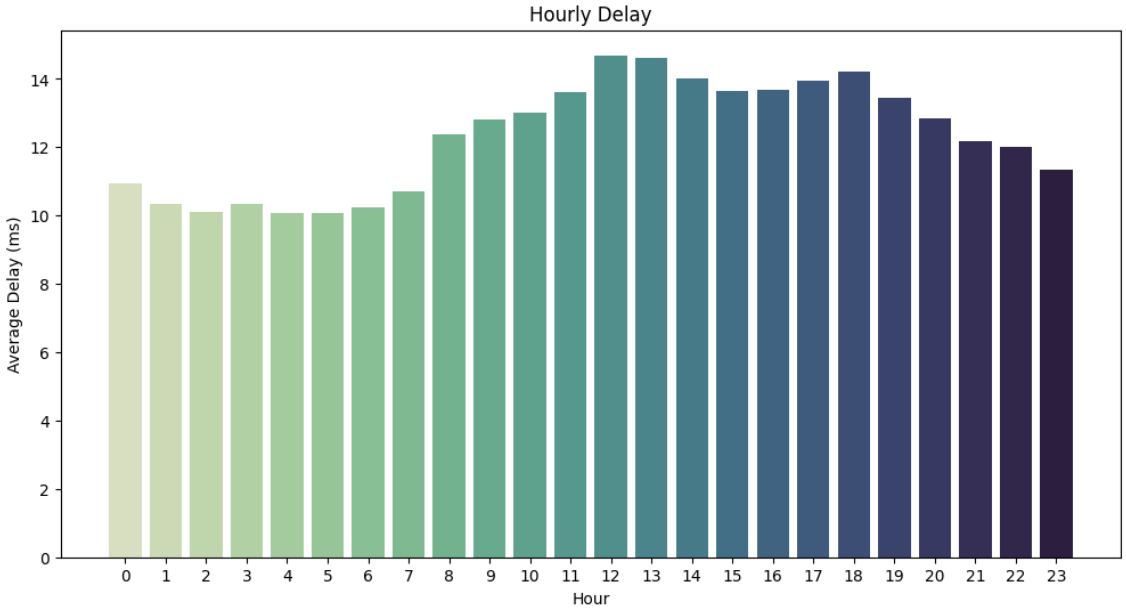


Figure 3.5: Average Hourly E2E Network Latency

The results indicate that the variation in E2E network latency follows the expected pattern of progressively increasing as the day progresses.

Conversely, as the day approaches its end, the values gradually decrease. These variations can be attributed to fluctuating network traffic and usage patterns among users throughout the day.

During periods of high activity, when many people access the network simultaneously, the increased demand leads to more traffic and, subsequently, higher delays. For these reasons, the peak of latency values, as indicated in the figure, occurs at 12 p.m. and 1 p.m., reaching an average of approximately 14 ms. On the other hand, the values are lower during the night and early morning hours due to fewer active users and, consequently, less congestion, as demonstrated in figure. These findings align with the results obtained in [84], which described similar behavior regarding network traffic.

Furthermore, Figure 3.6 provides additional insights into the hourly variation through a *boxplot*. A *boxplot* is a standardized method that summarizes the data distribution, including statistical measures such as the maximum, minimum, median and quartiles of the data [95]. However, the primary advantage of using a boxplot, in addition to displaying the previously mentioned statistical information, lies in its ability to identify outliers in the dataset.

Outliers are identified based on the Interquartile Range (IQR) concept. The IQR is a statistical measure of data dispersion, representing the range between the first and third quartiles of the data. In this case, an *outlier* is defined as any data point with at least 1.5 interquartile ranges below the first quartile or at least 1.5 interquartile ranges above the third quartile.

The choice of 1.5 as the scale is a widely accepted convention in many statistical analyses and visualizations. It finds a balance between being sensitive enough to detect potential outliers and avoiding the classification of normal variations in the data as outliers [96].

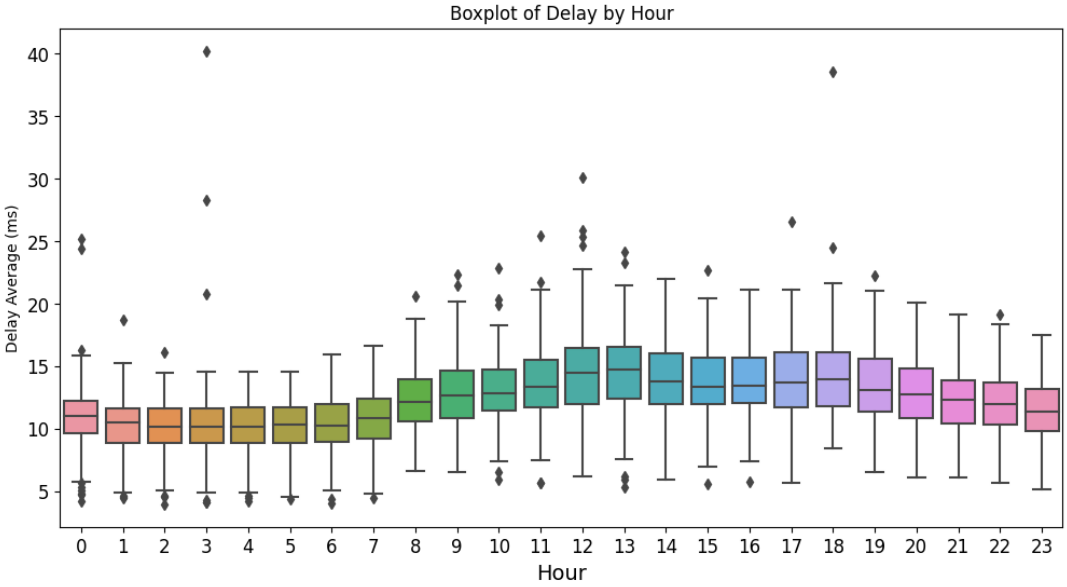


Figure 3.6: Average Hourly E2E Network Latency using Boxplots

Observing the Figure 3.6 reveals the maximum and minimum values and the presence of outliers associated with each hour of the day. These outliers, represented by points above and below the upper and lower whiskers, indicate abnormal measurements recorded at specific hours. In this data context, these anomalies can be attributed to synchronization issues with the measuring devices, network maintenance operations or unexpected events that impact the network's records.

Moreover, the difference between the maximum and minimum values and the IQR distance facilitates the analysis of data variability and dispersion. This analysis, in turn, aids in identifying potential factors that influence E2E latency delays at different times of the day, thereby contributing to optimizing system performance and minimizing latency issues.

Consistent with the results presented in the previous figure, the values of E2E network latency exhibit more fluctuations during the daytime than at nighttime. This pattern aligns with prior findings, suggesting that increased network activity and daily traffic volumes lead to heightened network congestion and, consequently, a higher likelihood of outliers occurring throughout the day.

The days of the week were also considered to conduct a more in-depth data-driven analysis of daily E2E network latency values patterns, as depicted in Figure 3.7. This approach enables the extraction of more complete insights into potential trends by examining the variations for each day, which can then also be used to optimize network performance.

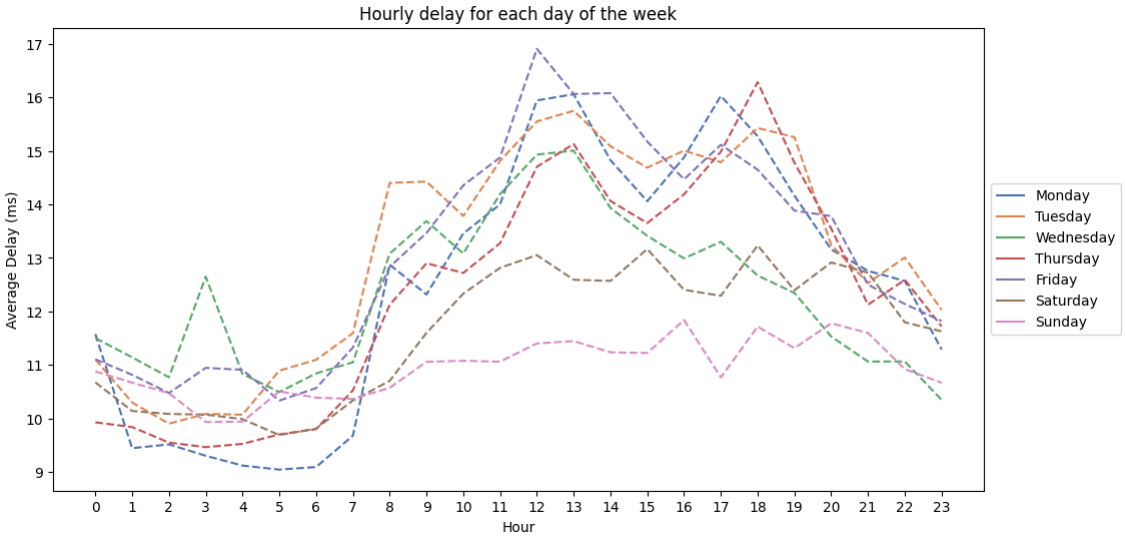


Figure 3.7: Average Hourly E2E Network Latency for each Day of the Week

The main conclusion drawn from this breakdown is that the day of the week does influence latency values throughout the day. The results also indicate that E2E network latency tends to be higher on weekdays compared to weekends.

This variation can be attributed to the substantial increase in usage during weekday work activities. As people engage in their daily routines, commute to and from work, and carry out other regular tasks, the demand for network resources significantly rises, resulting in increased delays in the network. In contrast, the reduced work-related activity during weekends leads to a slightly decreased demand on the network, which in turn contributes to fewer delays.

In addition, Figure 3.8 aims to complete the latest analysis by incorporating the respective standard deviations for each day of the week.

This inclusion enhances the explanation of the overall delay variation since the standard deviation exposes the variability in the data across different hours.

In line with previous findings, the standard deviation demonstrates lower values at night, indicating less variability in the data during those hours and, consequently, more stable measurements. During the day, the standard deviation increases, suggesting less stable measurements. Once again, these conclusions align with those derived from the previous analyses.

Abnormal peaks are also evident in the standard deviation plot. Several factors could have contributed to these occurrences, whereas the most likely reason involves failures in the experimental testbed or interferences in measurements, which had an impact on the results, as explained in previous sections.

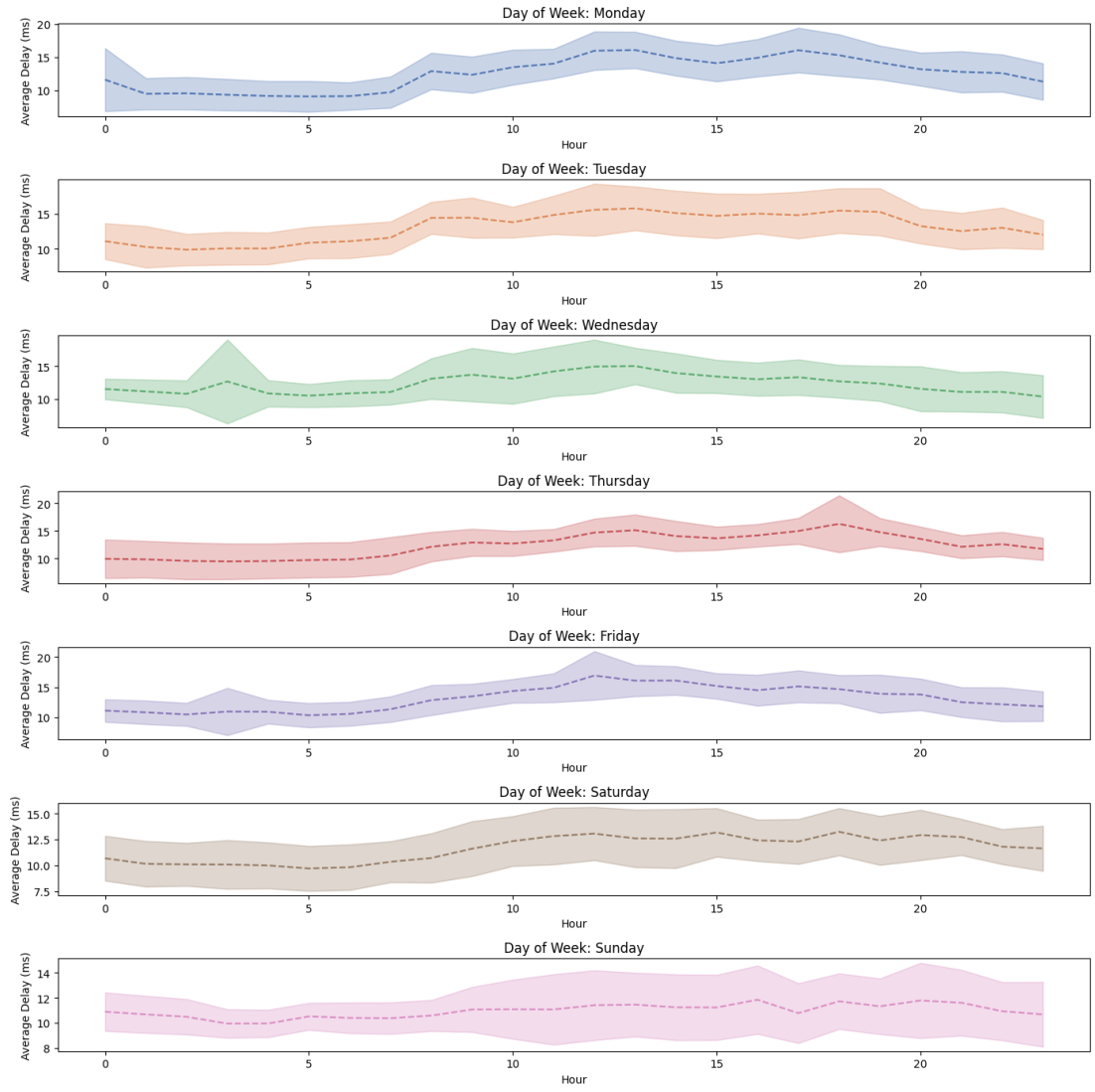


Figure 3.8: Average Hourly E2E Network Latency with Standard Deviation for Each Day of the Week

The analysis depicted in Figure 3.9 is intended to improve anomaly detection and offer a more precise visualization of how data is distributed across various hours on weekdays concerning E2E network latency.

Analyzing the figure highlights a difference in outlier occurrences between weekdays and weekends. This disparity is mainly attributed to the higher weekly user volume, resulting in more pronounced network congestion. It is consistent with the findings of earlier analyses.

Furthermore, the rise in traffic volumes during daytime hours also corresponds to an increased number of outliers compared to nighttime. This suggests a potential correlation between network congestion and latency spikes. Among the days of the week, Tuesday has the fewest outliers.

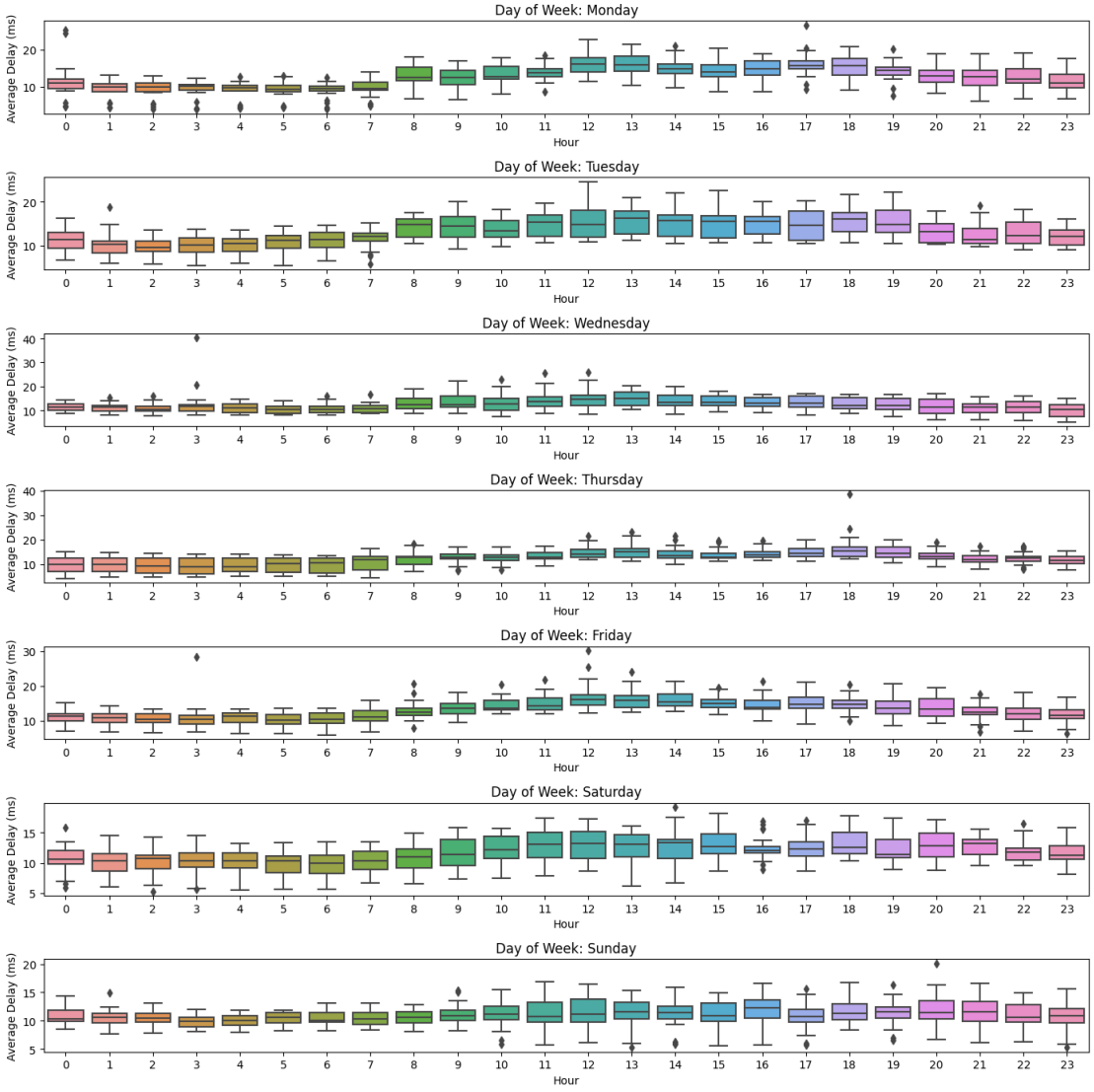


Figure 3.9: Average Hourly E2E Network Latency Delays for Each Day of the Week using Boxplot

In parallel with the previous analyses, an additional technique was explored to validate the remaining approaches' findings and provide other types of insights from the data, namely Kernel Density Estimation (KDE). It estimates a Probability Density Function (PDF), approximating the variable's underlying distribution smoothly [97].

The height of the curve at a specific point signifies the density of variable values in proximity to that point. Higher values denote regions where the variable is more concentrated, while lower values indicate areas with lower density.

This research utilizes Gaussian Kernel Density Estimation to estimate the Probability Density Function (PDF) of delay values. This method employs non-parametric techniques to smooth the data using Gaussian kernels.

The Gaussian kernel function is applied to each observed data point, creating a smooth curve around each point. These individual smoothed curves, represented as Gaussian distributions, are then combined through summation to derive the overall estimated density function. The Gaussian kernel's smoothness and flexibility enable an accurate representation of the data's density, effectively capturing local variations while preserving the overall shape of the distribution.

The following equation defines the Gaussian Kernel Function:

$$K(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3.2)$$

where μ represents the distribution's mean, σ denotes the standard deviation and x corresponds to a data point from the dataset.

In this study, the KDE was implemented using Sea born library's `kdeplot` method [98]. Thus, the results are visualized in Figure 3.10, which displays the KDE plot.

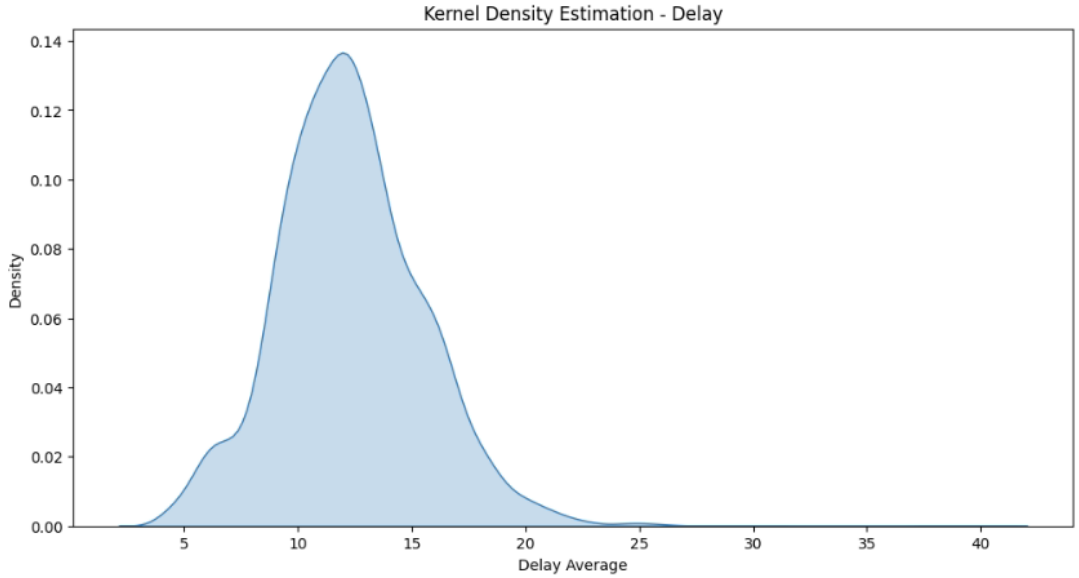


Figure 3.10: Kernel Density Estimation of E2E Network Latency for Each Day of Week

According to the figure, the E2E network latency values exhibit a higher density within the approximate range of [10, 14] ms, indicating that most delays fall within this interval.

Similar to the preceding analyses, this investigation also explores the impact of each day of the week on E2E latency delays, as illustrated in Figure 3.11.

The results reveal that weekdays exhibit higher latency values than weekends, with Tuesday and Wednesday standing out in particular. This observation could imply a greater likelihood of increased traffic on these specific days.

Conversely, there is a slight increase in density for lower E2E latency values on Mondays and Thursdays, suggesting a potential reduction in traffic congestion. This consistent pattern further underscores the distinction between weekday and weekend traffic patterns, underscoring the importance of targeted analysis and potential adjustments in traffic management strategies to optimize traffic flow.

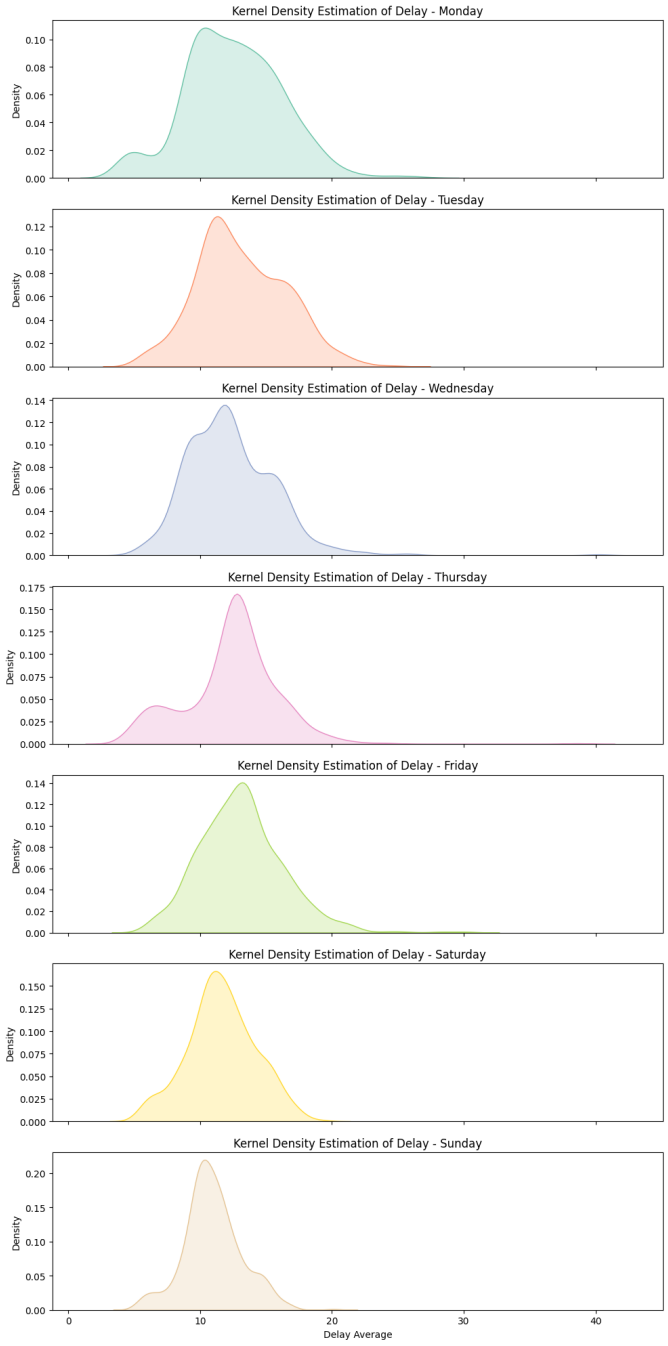


Figure 3.11: Kernel Density Estimation of E2E Network Latency for Each Day of Week

3.2 Data Splitting: Evaluation Methods

Dataset splitting is the stage at which the original dataset is divided into multiple subsets: training, validation and test. This separation permits an impartial evaluation of the model's ability to generalize to unobserved data. Without adequate data splitting, the model's performance and predictions could be unreliable. Additionally, data splitting helps identify and mitigate overfitting, which occurs when the model captures noise and specific patterns instead of learning general patterns.

There are different ways to split the data into training and evaluation sets. In the context of this project, three methods were evaluated, namely:

- K-Fold Cross-Validation;
- Times Series Split;
- Train-Validation-Test Split.

In the upcoming sections, these methods will be further detailed and compared to determine their effectiveness and contribution to the project.

3.2.0.1 K-Fold Cross-Validation

K-Fold Cross-Validation is a technique used to evaluate the model's performance. It addresses the limitations of a single train-validation-test split by providing a more complete evaluation of a model's generalization ability.

The basic idea behind this approach consists of dividing the original sample into K subsamples, also called *folds*. Subsequently, the model is trained and evaluated K times. In each iteration, a different fold performs as the validation set, while the rest are used as the training set, as illustrated in Figure 3.12. The K results from the folds are then averaged to produce a more robust performance estimation [99].

One of the main advantages of this procedure is its ability to minimize the risk of model overfitting. Through the evaluation of the model on multiple validation sets, it offers a more comprehensive understanding of its performance on unobserved data. Furthermore, it uses all available data for training and validation, making it a more data-efficient approach than traditional validation techniques.

Additionally, cross-validation facilitates hyperparameter optimization and model selection by comparing the performance of various models across different validation sets [100].

In the course of the project, each fold comprises the measurements from one week. When complete weeks are not available, new folds are generated.

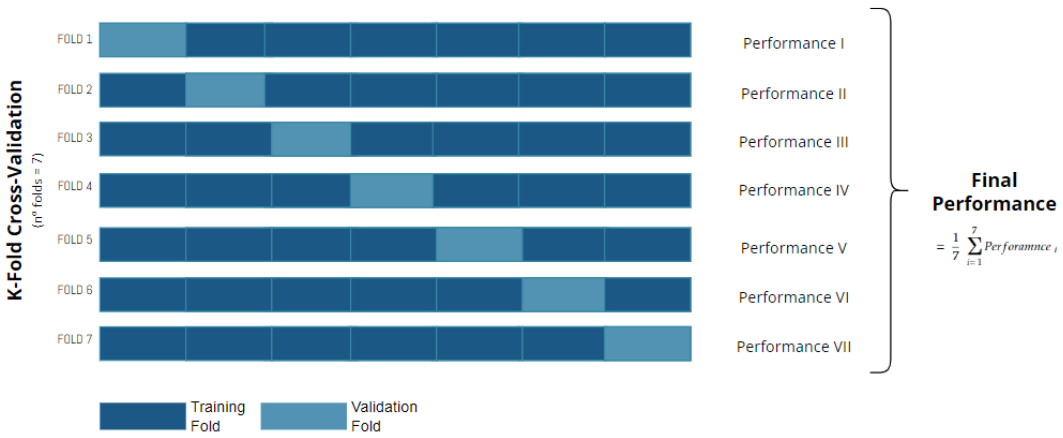


Figure 3.12: Illustration of the K-Fold Cross-Validation procedure

Furthermore, to prevent random shuffling of the data and, consequently, to obtain more realistic evaluations and more accurate analysis, each fold is constructed using consecutive events with the aim of preserving the sequential dependency of the measurements.

3.2.1 Time Series Split

Time Series Split is a cross-validation technique designed for time-series data [101]. It divides the data into a specified number of folds, similar to K-Fold Cross-Validation. However, in Time Series Split, the data's chronological order is maintained, and each fold is generated in a manner that preserves the temporal sequence of data points.

This is crucial in time-series analysis because it guarantees that the model is trained on historical data and assessed on future data, simulating real-world scenarios where predictions rely on past information.

Figure 4.10 provides a visual representation of this technique's process, where the data is partitioned into multiple folds, each representing a distinct time period. As illustrated in the figure, in contrast to the previous technique, which used all available data for training and testing, this approach incrementally incorporates more folds for model training. However, the final performance is computed by averaging the results from all folds.

Similar to the preceding approach, each fold includes measurements from one week. Additionally, in situations where the total number of measurements for a week does not match, new folds are established. This process is designed to maximize data dependencies and capture patterns or trends to achieve more precise outcomes.

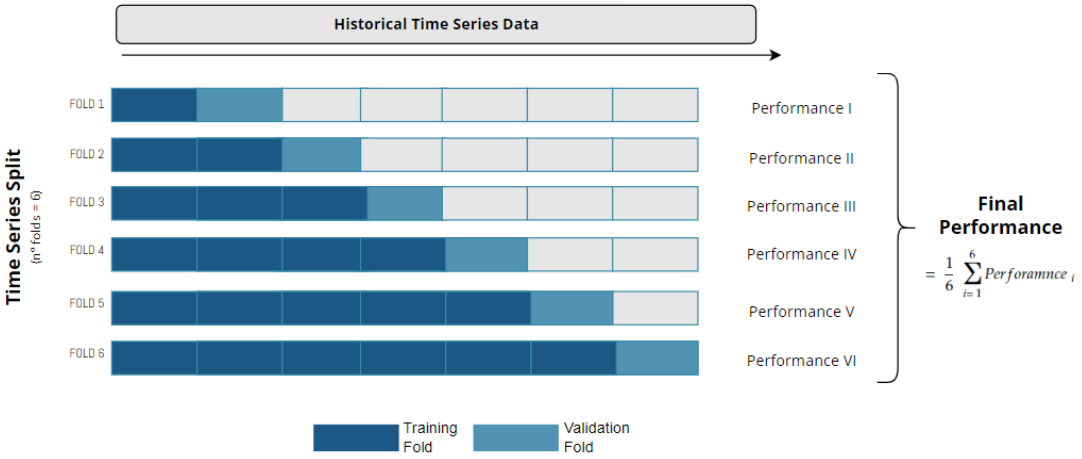


Figure 3.13: Illustration of the Time Series Split procedure

3.2.2 Train-Validation-Test Split

The Train-Validation-Test Split is a data partitioning technique which divides the dataset into three distinct subsets:

- **Training Set**

  The greatest subset used to train the ML model is the training set. The model learns patterns and relationships from this data and adjusts its parameters to minimize the objective function. Furthermore, this subset must represent the diversity and variability of the overall dataset to maximize the model performance and be large enough to generate meaningful results.

- **Validation Set**

  This subset is used to fine-tune the model's hyperparameters and select the best-performing model based on its performance during validation. Evaluating a trained model on the validation set contributes valuable insights into its ability to generalize to unseen data, identifying potential issues like overfitting, which can significantly impact the model's performance in real-world scenarios.

- **Test Set**

  The Test set remains independent of the other subsets and evaluates a trained model's ability to perform on unseen data. This evaluation offers insights into whether the model has effectively learned meaningful patterns and can make accurate predictions in real-world scenarios. This assessment is crucial for determining the model's reliability and effectiveness.

Accurately allocating data to each subset is crucial to prevent the model from overfitting and ensure a more realistic and accurate model evaluation.

3.3 Model Selection

In this section, the models used to evaluate E2E latency delay predictions are described. The selected models enclose a variety of approaches, spanning from conventional machine learning algorithms to deep neural networks, including LSTM. The selection of ML models was based on their applicability to the issue and their efficacy in analogous scenarios.

3.3.1 ML Algorithms

Once the primary purpose of the FLOYD Project is to lead V2X communications for platooning, reducing communication delays becomes vital for achieving seamless and safe coordination among platooning vehicles, contributing to increased efficiency and overall system reliability.

In this way, ML algorithms can significantly impact building an intelligent system capable of predicting latency in real-world scenarios to ensure reliable communication and coordination between platooning vehicles.

The ML algorithms evaluated to accomplish the project's objectives were [62] [74]:

- Extreme Gradient Boosting (XGBoost);
- Light Gradient Boosting Machine Regressor (LGBMR);
- Random Forest (RF);
- Support Vector Regressor (SVR).

These algorithms were implemented using the `scikit-learn` Python library [102], which provides various tools for development and implementation. Further details about each model can be found in the sections below.

3.3.1.1 Extreme Gradient Boosting

Extreme Gradient Boosting [103] is an ensemble learning method that combines multiple weak models to create a more robust one. More specifically, this algorithm is based on gradient boosting, where multiple decision trees are trained iteratively to correct the residuals from the previous ones, ultimately creating a strong and more accurate model.

The XGBoost algorithm introduces several enhancements to the original gradient boosting method introduced by Friedman in [104].

Beyond using a more efficient tree construction algorithm, it incorporates Lasso (L1) and Ridge (L2) regularization techniques in its objective function to control model complexity and reduce variance. Additionally, it employs parallel processing capabilities to accelerate the training process, making this model robust and efficient for handling large datasets.

Another of the key enhancements of this method is its ability to handle missing data effectively. It can automatically learn how to handle missing values during the training process best, reducing the need for data preprocessing. Moreover, it contains an extensive selection of hyperparameters that can be adjusted to optimize model performance and meet the requirements of specific problems.

Together, these abilities make XGBoost one of the most robust and popular algorithms used in supervised learning.

3.3.1.2 Light Gradient Boosting Machine Regressor

Light Gradient Boosting Machine Regressor is another gradient-boosting framework that uses decision trees to increase the model's efficiency and reduce memory usage [105]. It distinguishes itself from other boosting algorithms by utilizing a leaf-wise tree-splitting strategy instead of the conventional level-wise approach. This strategy selects the leaf that most reduces the loss for tree expansion. This algorithm applied two techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) [105]. These techniques resolve the limitations of the histogram-based algorithm that is predominantly implemented in all Gradient Boosting Decision Tree (GBDT) frameworks.

The main objective of using GOSS is to indicate the importance of data instances in order to select ones with higher gradients to compute the information gained. It excludes the remaining ones, as they are already well-trained. However, this approach presents a bias problem towards the samples with more significant gradients and changes the original data distribution.

In order to address this problem, GOSS random sampling on the data instances exhibiting small gradients while preserving all the samples displaying high gradients. Given that the sample remains skewed towards data exhibiting high gradients, the computation of information gain involves augmenting the weights of data instances characterized by small gradients by adding a constant multiplier [106].

Furthermore, LGBMR uses EFB to reduce the number of features while retaining only the most informative ones by combining mutually exclusive features nearly losslessly.

3.3.1.3 Random Forest

Random Forest, proposed by L. Breiman [107], is a supervised learning algorithm that uses ensemble learning to create a model by training multiple decision trees on different data subsets.

Initially, it randomly creates multiple subsets of the training dataset through bootstrapping. This method randomly selects samples with replacements from the original dataset to ensure data variability in the defined subsets. Then, for each subset, a decision tree is constructed. Each decision tree is constructed recursively by selecting, at each node, the optimal division based on a criterion, as illustrated in Figure 3.14.

In addition to randomizing the data, this model introduces randomness in feature selection. At each split in the decision tree, a random subset of features is considered for splitting, reducing the correlation between individual trees and making the ensemble more robust [108]. Furthermore, instead of considering all features for splitting at each decision tree node, a random subset of features is chosen. This approach further reduces the correlation between individual trees and makes the model less prone to overfitting.

The tree stops splitting once a predefined stopping criterion is reached, such as attaining a maximum depth or having a minimum number of samples in a leaf node. After each decision tree is trained, it can be used for making predictions. In classification tasks, each tree votes for the class it predicts, with the class receiving the most votes used as the final prediction. For regression tasks, the final prediction is typically the average of the predictions from all the trees [108]. This averaging process helps reduce variance and enhances the model's generalization performance.

Furthermore, this model is also suitable for high-dimensional data modelling because it can handle missing values of different data types [108]. By harnessing these capabilities, RF becomes a robust algorithm for various tasks.

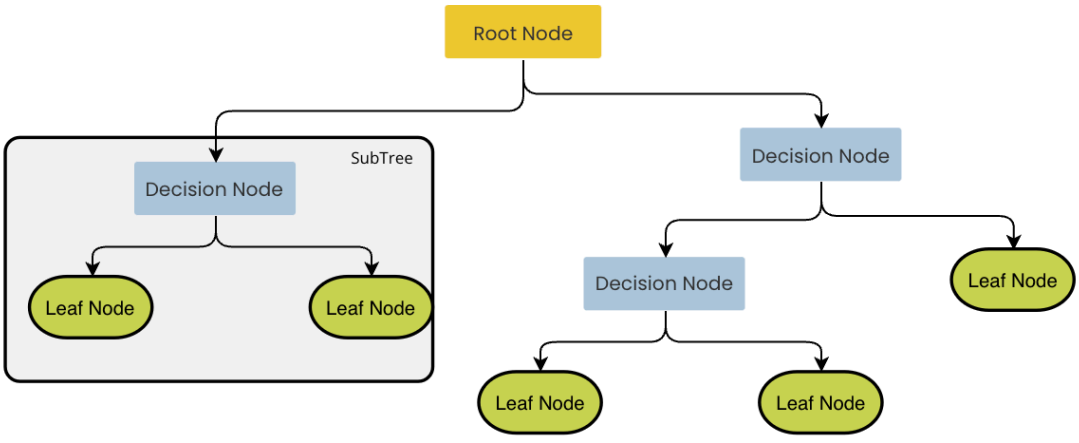


Figure 3.14: Decision Tree Structure [37]

3.3.1.4 Support Vector Regressor

Support Vector Regressor is a extension of Support Vector Machine (SVM) proposed in 1997 by Vapnik, Steven Golowich and Alex Smola [109] to performing regression problems.

The principle behind SVR is to define a hyperplane that maximally separates the data points, as Figure 3.15 illustrates. Unlike traditional regression methods, SVR aims to minimize the error within a certain margin around the hyperplane. This hyperplane works as a decision boundary and is determined by selecting a subset of training samples, known as support vectors, which corresponds to the data points closest to the hyperplane.

Maximizing the margin allows the hyperplane to be fitted equidistantly from the boundaries, allowing for tolerable errors. A more significant margin corresponds to a reduced risk of the model overfitting. Instances that fall beyond the acceptable margin are assigned weights proportional to their distances from the boundaries. Consequently, there is a linear increase in penalties for such errors. This mechanism provides greater flexibility in capturing complex relationships between variables, all while maintaining robustness against outliers. In the context of this model, an outlier refers to any data point that lies outside the defined margin.

A key feature of SVR is the use of kernel functions. These mathematical functions transform the data into a higher-dimensional space, enabling it to establish a linear boundary in a non-linear space, thus making it suitable for non-linear regression tasks [110]. The main kernel functions are Linear, Non-Linear, Polynomial, Radial Basis Function (RBF) and Sigmoid.

Overall, SVR uses flexibility to define the level of acceptable error in the model by defining a hyperplane that fits the data [111].

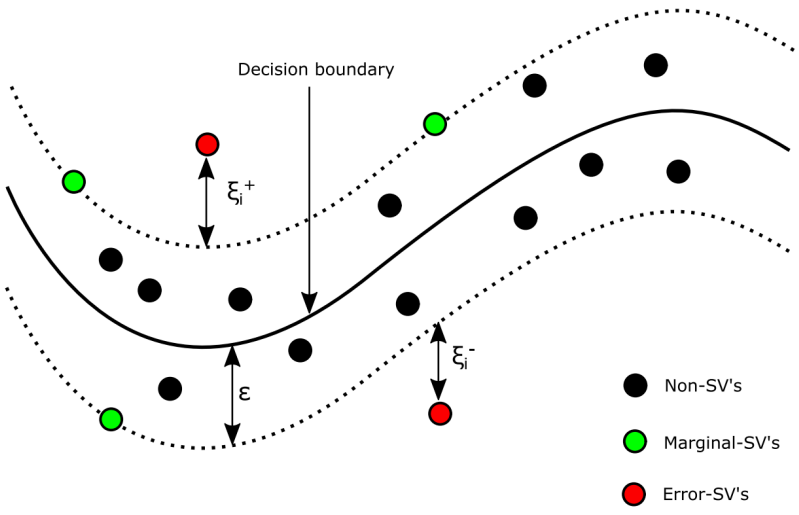


Figure 3.15: Structure of SVR [112]

3.3.2 Deep Learning

Surveys conducted in network communications have indicated promising results for Deep Learning, which can significantly enhance the responsiveness and performance of networked applications requiring real-time interactions, like autonomous vehicles.

This work implemented LSTM and MLP architectures to attempt to understand the data's temporal dependencies and complex patterns.

3.3.2.1 Long Short Term Memory

Long Short-Term Memory networks, a specific type of RNN architecture, are designed to mitigate the vanishing and exploding gradient problems in traditional RNNs, making them better suited to capturing long-term dependencies in sequential data.

They were introduced in [59] and consisted of memory cell units that enabled data retention for extended periods. Three gates manage information flow into and out of the cell: the input gate, the output gate and the forget gate, as shown in Figure 3.16. The forget gate is responsible for filtering which information from the previous cell state needs to be memorized and what should be discarded as it is no longer helpful, while the input gate controls the flow of information entering the cell state [58]. Finally, the output gate determines and regulates the outputs. These gates, working together, allow the network to selectively retain or forget information, making it highly effective in processing and generating complex patterns.

In this research, different combinations of hyperparameters for LSTM were tuned to select the combination with the lowest validation error. These hyperparameters include the number of LSTM layers, learning rate, momentum and batch size.

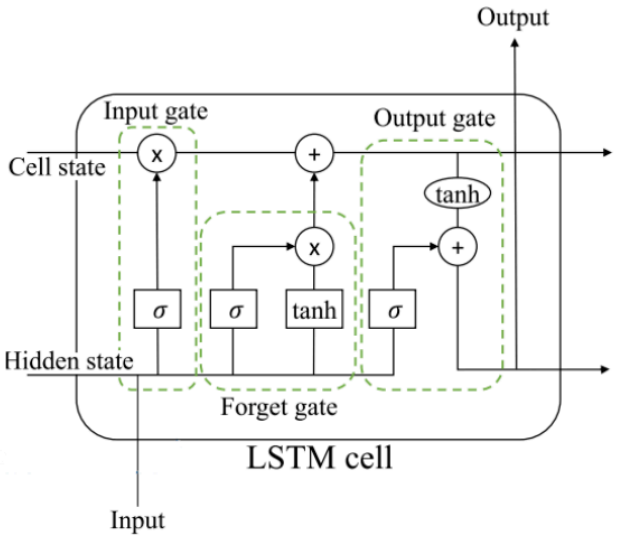


Figure 3.16: Structure of LSTM cell [67]

3.3.2.2 Multi-Layer Perceptron

The Multi-Layer Perceptron is a feedforward Artificial Neural Network (ANN) composed of multiple layers of interconnected perceptrons [51].

The MLP architecture comprises an input layer, one or more hidden layers and an output layer, as shown in Figure 3.17. The number of hidden layers and the number of neurons in each layer depend on the problem's complexity. The input layer receives the data, which is then processed through the hidden layers. Each hidden layer applies a non-linear activation function to transform the input and extract relevant features. The output layer produces the final outputs based on the learned patterns in the preceding layers.

Furthermore, the activation function used in the output layer depends on the nature of the problem, such as sigmoid for binary classification or softmax for multi-class classification [113].

This neural network learns to optimize its weights and biases through a procedure called backpropagation [114], where errors are propagated backwards from the output layer to adjust the parameters in each layer.

The depth and width of the MLP architecture can be adjusted to balance model complexity and computational efficiency.

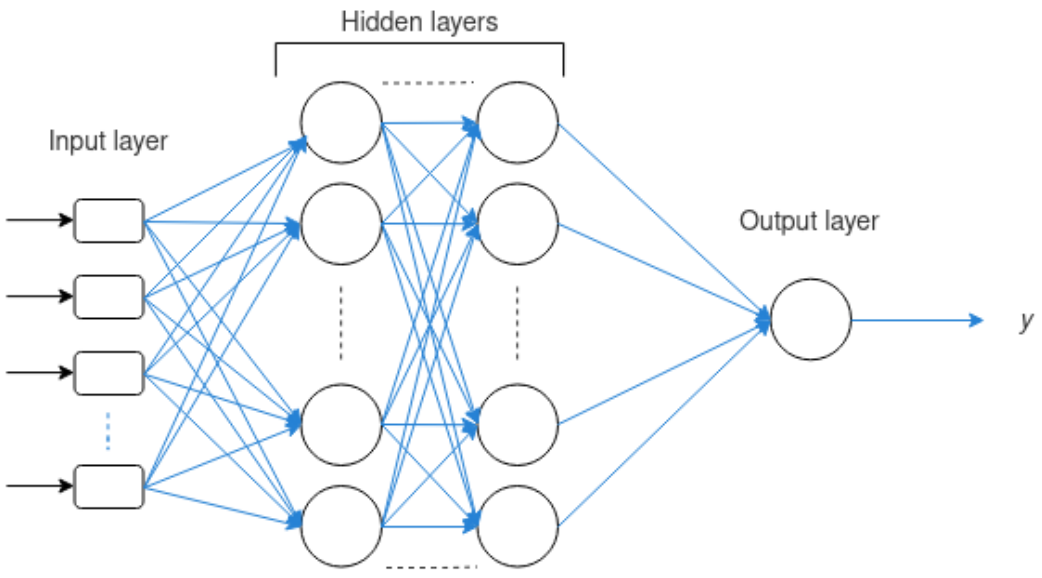


Figure 3.17: MLP Architecture [115]

3.4 Model Evaluation: Performance Metrics

In the context of this research, the selected metrics for evaluating the model's performance were:

- Coefficient of Determination (R^2);
- Mean Absolute Error (MAE);
- Root Mean Square Error (RMSE).

These metrics are commonly employed in QoS methodologies to assess the precision of forecasts and the effectiveness of model fitting. Additional information regarding these metrics can be found in the following sections.

In the mathematical formulas that follow, *n* represents the total number of observations, y_i denotes the observed value, $\hat{y}_i$ represents the predicted value and $\bar{y}$ indicates the mean of the sample.

3.4.1 Coefficient of Determination

The coefficient of determination, also known as R-square (R^2), measures how well a statistical model explains the variation in the dependent variable based on independent variables. According to its capacity to relationship between independent and dependent variables, R-squared can take values in the range [-∞, 1] [116].

A coefficient of determination equal to 1 indicates that the prediction model perfectly accounts for the variance in the dependent variable. Thus, the closer the values are to 1, the more accurately the prediction model performs.

Conversely, R^2 can also take on negative values, indicating that the regression model performs worse than a model that simply predicts the mean of the dependent variable. This situation arises when the model fails to capture any discernible patterns or trends in the data, essentially rendering its predictions random.

The formula used to calculate this metric is:

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \quad (3.3)$$

where SSR is the Sum of Squared Residuals, which represent the unexplained variance in the dependent variable [84]. Conversely, SST means the Total Sum of Squares and measures the total variance in the dependent variable [84].

3.4.2 Mean Absolute Error

The Mean Absolute Error (MAE) represents the average magnitude of the absolute difference between ground truth and predicted values [117].

As demonstrated by Equation 3.4, the MAE is computed by adding the absolute differences between each observed value and its corresponding predicted value, and then dividing the sum by the total number of data points.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (3.4)$$

A lower MAE value denotes that the model exhibits a smaller average error when predicting the dependent variable, which illustrates better model performance. This metric is not affected by outliers in the data, representing a difference for the remaining ones.

3.4.3 Root Mean Square Error

The Root Mean Square Error (RMSE) is commonly used in supervised learning problems since it represents the square root of the average squared difference between the predicted and true values [117]. A smaller RMSE indicates more accurate predictions, while a higher RMSE suggests greater error and less precision.

Equation 3.5 defines the RMSE calculation formula. In this equation, the first step involves computing the square of the differences between predicted and true values. Then, the average of these squared differences is calculated, resulting in the Mean Square Error (MSE) (Mean Squared Error).

Finally, the square root of the MSE is determined, yielding the RMSE. Once this metric involves squaring errors before averaging, it penalizes significant errors more severely, making it sensitive to outliers.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (3.5)$$

Similarity to MAE, RMSE presents results in the same units as the dependent variable, facilitating interpretation.

Chapter 4

Results and Discussion

This chapter introduces the implemented method used for predicting network latency values and thoroughly analyzes the results of each experiment. The implementation of various evaluation methods aims to assess the model's performance more effectively and understand its strengths and limitations. Additionally, this chapter addresses a paper accepted at one conference as part of the internship.

4.1 Contributions

During the internship period within the scope of the FLOYD project, a paper was submitted and accepted at the 24th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) Conference. This conference focused on wireless networking technologies and their pivotal role in future network communication systems.

The submitted paper [7] contributed to validating an E2E 5G orchestration architecture proposed in the FLOYD project. This architecture was designed to monitor RAN data and leverage the capacity of ML algorithms to optimize network performance by dynamically allocating network resources based on real-time data analysis.

Furthermore, this work demonstrates the potential to significantly advance wireless networking technologies by harnessing the power of AI to develop more intelligent and autonomous networks in the future. These advancements ultimately improve user experiences, including those related to autonomous driving.

4.2 Predictive Task Definition

As discussed in prior chapters, the data used in this study consists of time series data, representing a sequential record of observations denoted as $Y = \{y_i, y_{i+1}, \ldots, y_t\}$. Here, y_i corresponds to the value of Y at time i and t denotes the length of the time series gathered over a period. Given the study's emphasis on time series forecasting, the approach utilized is strictly autoregressive modeling, wherein future values are predicted using its historical lags.

For this effect, a set of observations was constructed based on past data values, as Figure 4.1 indicates. Each observation comprises a feature vector ($\{y_i, y_{i+1}, \ldots, y_t\}$), denoting the previous p values, and a target vector ((y_{p+1})), representing the value to predict. This approach assumes that are no time dependencies larger than *p* [118].

$$Y_{[n,p]} = \begin{bmatrix} y_1 & y_2 & \cdots & y_{p-1} & y_p \\ \vdots & \vdots & \vdots & \vdots & \\ y_{i-p+1} & y_{i-p+2} & \cdots & y_{i-1} & y_i \\ \vdots & \vdots & \vdots & \vdots & \\ y_{t-p+1} & y_{t-p+2} & \cdots & y_{t-1} & y_t \end{bmatrix} \quad \begin{bmatrix} y_{p+1} \\ \vdots \\ y_{i+1} \\ \vdots \\ y_{t+1} \end{bmatrix}$$

Figure 4.1: Sliding Window method [118]

Furthermore, various window sizes were considered to determine the optimal number of previous values to include in the feature vector. In this study, the range of past values employed varies from the most recent moment ($p = 1$), representing the last 15 minutes, to the day before the moment of prediction ($p = 96$).

Table 4.1 summarizes the window sizes considered for this analysis.

Window Sizes	
p	Time
1	15 Min
2	30 Min
3	45 Min
4	60 Min
6	90 Min
8	120 Min
12	180 Min
24	360Min
48	720 Min
72	1080 Min
96	1440 Min

Table 4.1: Window Sizes considered for the study

The goal of experimenting with multiple time windows is to balance capturing enough historical information for accurate predictions while mitigating two potential issues: the introduction of noise with more oversized time windows and the risk of insufficient information to capture data patterns.

Then, the final objective is to build a model $f : X \rightarrow Y$, where f represents the regression function, capable of capturing temporal dependencies and, subsequently, accurately predicting future values based on historical information.

4.3 Results

In this section, the results of the performance of the models are presented and discussed for each of the methodologies followed in order to determine if there are any significant differences in their performance. Therefore, these analyses aim to identify optimal practices and areas for further investigation and enhancement.

4.3.1 K-Fold Cross-Validation

This methodology divides the data into multiple folds and the final performance is evaluated by averaging the performance of each fold. This procedure assures that the model is trained and tested on different subsets of the dataset, ensuring a more rigorous evaluation of its performance. It also aids in identifying any potential overfitting or underfitting issues that may arise during the training process.

Throughout the project, it was established that each fold should include measurements collected over one week. From the data that was accessible, a total of seven folds were established. Five of these folds consisted of comprehensive measurements from a single week, while the remaining two incorporated data from incomplete weeks, as depicted in Figure 4.2.

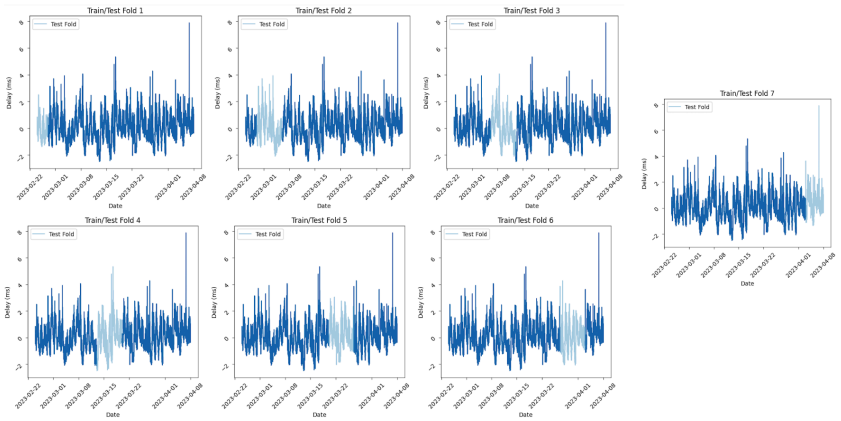


Figure 4.2: Train and Test Folds - K-Fold Cross-Validation

Since K-Fold Cross-Validation is a methodology not originally designed for time series data, this division arises to preserve the data's temporal dependencies during the model training process. This procedure empowers the model to recognize patterns and make predictions based on the sequential nature of the data. Consequently, this leads towards more dependable outcomes.

Throughout this approach, numerous models and varied window sizes were tested to find the optimal combination for accurately predicting future data points, as outlined in Sections 3.3 and 4.2, respectively.

The results of these experiments demonstrate that the window sizes impacted the model's outputs, with larger window sizes generally resulting in more accurate predictions. However, it was also observed that excessively large window sizes could lead to overfitting and decreased performance.

Among the time windows assessed, detailed in Section 4.2, the highest-performing ones are the four following: 45 minutes, 60 minutes, 90 minutes, and 120 minutes. Within this subset, the models demonstrating superior performance are MLP and SVR, with MLP showing a slight advantage over SVR.

These performance comparisons are illustrated in Figure 4.3.

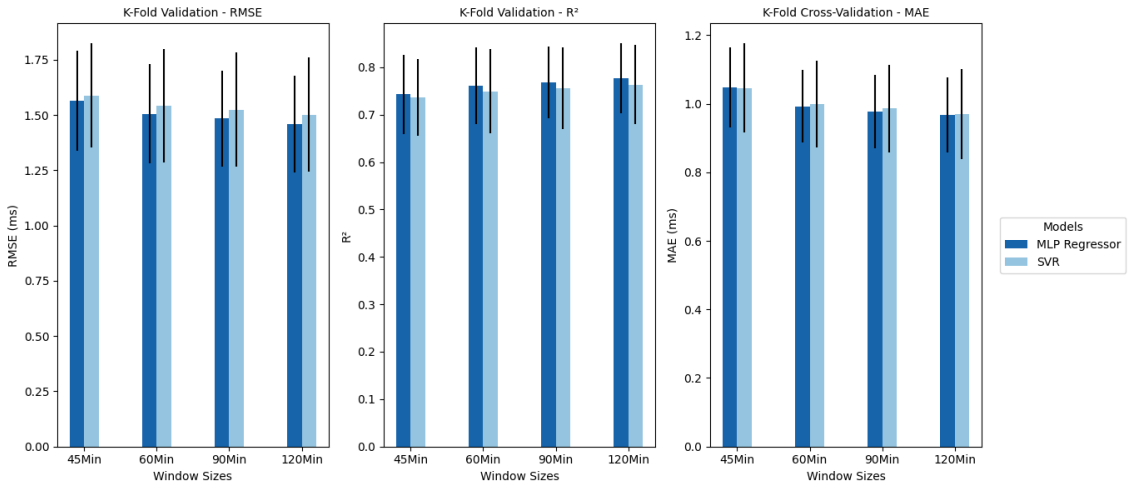


Figure 4.3: Result of K-Fold Cross-Validation Approach

Based on the depicted results, all-time windows show very similar performance, except for the 120-minute time window, which slightly outperforms the others. Regarding the models, MLP demonstrates a slight advantage over SVR. This finding meets with the conclusions from the literature review, where neural networks were expected to exhibit superior performance compared to conventional ML algorithms.

Delving deeper into the interpretation of the results, especially concerning the MAE, reveals that, on average, the predictions differ from the ground truth by approximately one millisecond across these window sizes. This indicates promising findings.

Furthermore, considering the coefficient of determination (R^2) values, which enable understanding the model's capacity to explain variations in latency delays based on past values, suggests that both models effectively capture the underlying patterns and relationships in the data, as their values hover around 80%.

Additionally, the results of the remaining ML algorithms tested indicate a clear tendency towards overfitting across nearly all the time windows examined in this study. In turn, it suggests that these models have become overly tailored to the training data, capturing noise and specific patterns that do not generalize effectively to unseen data.

The most plausible explanation for it relates to the model's complexity compared to the available training data. If the model is too complex, it might quickly memorize the training examples instead of learning the underlying patterns. As a result, the model performs poorly when faced with new, unseen data. Thus, balancing the degree of model complexity and the quantity of available information is pivotal. Other factors that could influence this model's behavior are noises in the data, which are highly possible due to the limitations of the implemented testbed and the numerous interferences detected when gathering measurements from the network.

One solution to minimize this problem is to expand the available data. With greater availability of information, the model will have more instances to learn from and better generalize its understanding of deeper patterns. Another approach is to simplify the model's complexity by adjusting hyperparameters or using regularization techniques. These techniques apply penalty terms to the cost function of the model, striking an equilibrium between model performance and model complexity.

The optimal configuration of a model is determined by hyperparameter tuning, which is a fundamental step in ML. Some commonly used techniques include Grid Search, Random Search [119] and Bayesian optimization. These approaches iteratively explore different values of hyperparameters to find the best combination that maximizes its performance. In this approach, two of them were applied to mitigate overfitting and improve the model's precision: Grid Search and Bayesian optimization.

XGBoost is one of the models susceptible to overfitting. Thus, the goal is to optimize its hyperparameters using the Bayesian optimization algorithm. This model choice relates to its performance in articles that propose predicting network attributes, namely on [70]. Additionally, it incorporates regularization techniques, including L1 and L2, to mitigate overfitting.

In this case, Bayesian optimization was employed because it efficiently explores the parameter space and makes informed decisions based on prior evaluations. It uses a probabilistic surrogate model to depict how the objective function behaves across the hyperparameter space. This allows the method to intelligently navigate the hyperparameter space by selecting configurations likely to yield the minimum of the objective function.

The choice of configuration is determined by balancing exploration and exploitation. Exploration involves searching for new and uncharted areas of the hyperparameter space, while exploitation focuses on promising regions that have shown strong performance in prior evaluations.

This approach was implemented using the `Optuna` Python library [120]. It consists of an automatic hyperparameter tuning framework designed for ML models.

The core concepts within this framework are the *study* and *trial*. A *study* relates to optimization process, including defining the objective function and hyperparameter search space. Conversely, a *trial* represents an execution of a single model run with a specific set of hyperparameters.

`Optuna` employs a Bayesian optimization algorithm called Tree-Structured Parzen Estimator (TPE) [121], which iteratively uses a history of *trials* to create a probabilistic model [120]. This model is then used to suggest the next set of hyperparameters for evaluation.

In each trial, TPE fits two Gaussian Mixture Models (GMMs) to the parameter values: one, denoted as $l(x)$, is fitted to the parameter values associated with the best objective values, while the other, denoted as $g(x)$, is fitted to the remaining parameter values [122]. The algorithm selects the parameter value x that maximizes the ratio $l(x)/g(x)$.

Figure 4.4 illustrates the progression of the objective function during the study, demonstrating how it changed as various configurations were investigated. In this study, the objective function to minimize is RMSE.

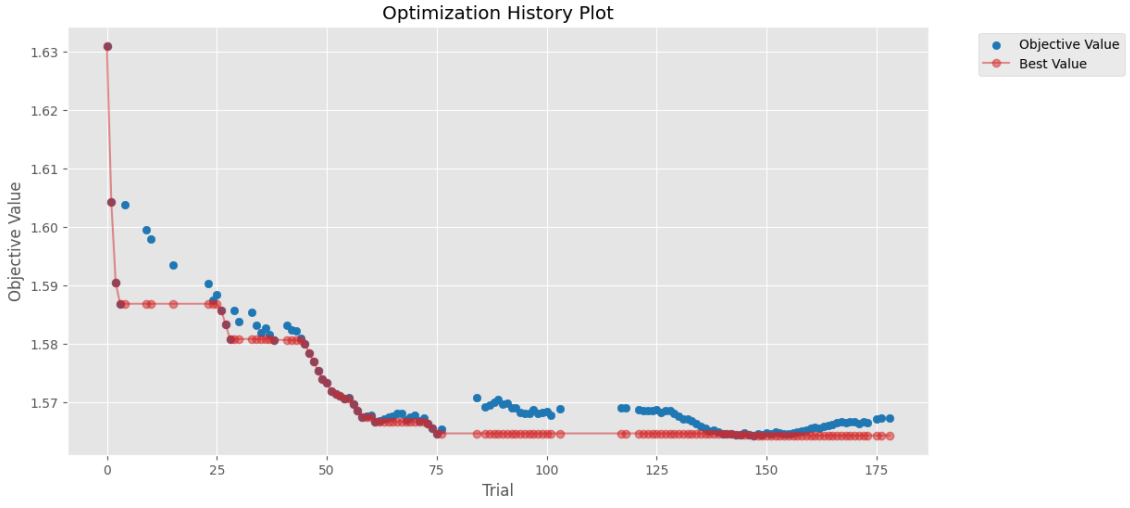


Figure 4.4: *Trials* Evaluation

The model underwent iterative tuning until the objective function reached its minimum value, as indicated by the figure's red points. Moreover, unpromising trials were efficiently pruned using `Optuna`'s functionalities, explaining the absence of specific objective function values in this figure. This approach minimizes unnecessary consumption of computational resources and expedites the convergence toward optimal results.

Several pruning algorithms are available, but according to the library's documentation, the most suitable one when implementing the Tree-TPE algorithm is the `Hyperband` algorithm [123].

This algorithm functions as a search space pruner, utilizing a combination of random sampling and early stopping to explore the hyperparameter space [124] efficiently. It gradually allocates more resources to promising trials while promptly discarding unpromising ones.

The outcomes of this optimization process are presented in Table 4.2, which compares the baseline model and the results of the optimized model.

Model		**MAE (ms)**		**R-Square**		**RMSE (ms)**	
		μ	σ	μ	σ	μ	σ
XGBoost **OVF**	Train	0.32347	0.01603	0.98194	0.00154	0.44671	0.02161
	Test	1.10441	0.13904	0.72287	0.08506	1.63144	0.25868
XGBoost **OPT**	Train	0.97593	0.1084	0.80199	0.0854	1.47135	0.19291
	Test	**1.02833**	0.12459	**0.77416**	0.07415	**1.51697**	0.23382

Table 4.2: Results of XGBoost Optimization Process

The model's tendency to overfit becomes evident when analyzing the results of the XGBoost OVF model, which displays a significant performance gap between its performance on the training and test sets. This behavior suggests that the model is overly tailored to the training data and fails to generalize effectively to unseen data. Another aspect demonstrating this behavior is the lower variance compared to that observed in the test set.

Subsequently, following the optimization process, perceptible improvements in the model's performance are observed, as evidenced by the outcomes of XGBoost OPT.

These results showcase that the model generalizes better to unseen data and is less susceptible to overfitting. However, its performance is still behind that of MLP and SVR for the same time window (120 minutes).

Furthermore, `Optuna` provides various visualization tools to analyze the results of individual trials and gain insights into the model's performance across different hyperparameter configurations.

Figure 4.5 illustrates the relationship between the values of each hyperparameter used during the study and their respective impact on the cost function. This representation aids in comprehending how each parameter's value influences the optimization process. From the figure, it can be concluded that higher values of *reg_alpha* result in higher values for the objective function.

The impact of each hyperparameter on the objective value was also computed, as shown in Figure 4.6. This analysis facilitates identifying which ones contribute the most to minimizing the objective value.

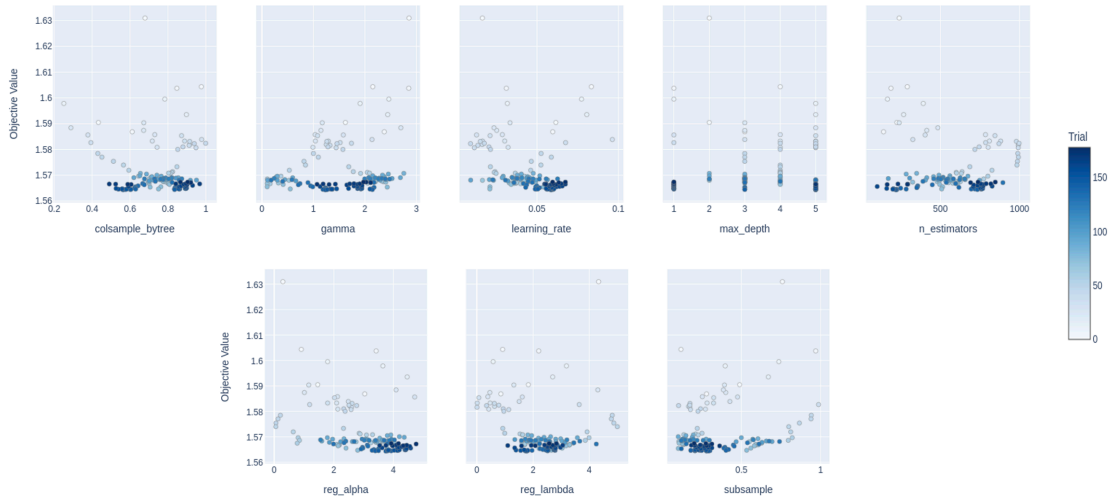


Figure 4.5: Individual Hyperparameter Performances from `Optuna`

In this instance, *gamma* had the most significant impact, followed by *colsample_bytree* and *learning_rate*. Conversely, *max_depth* and *n_estimators* had an insignificant impact.

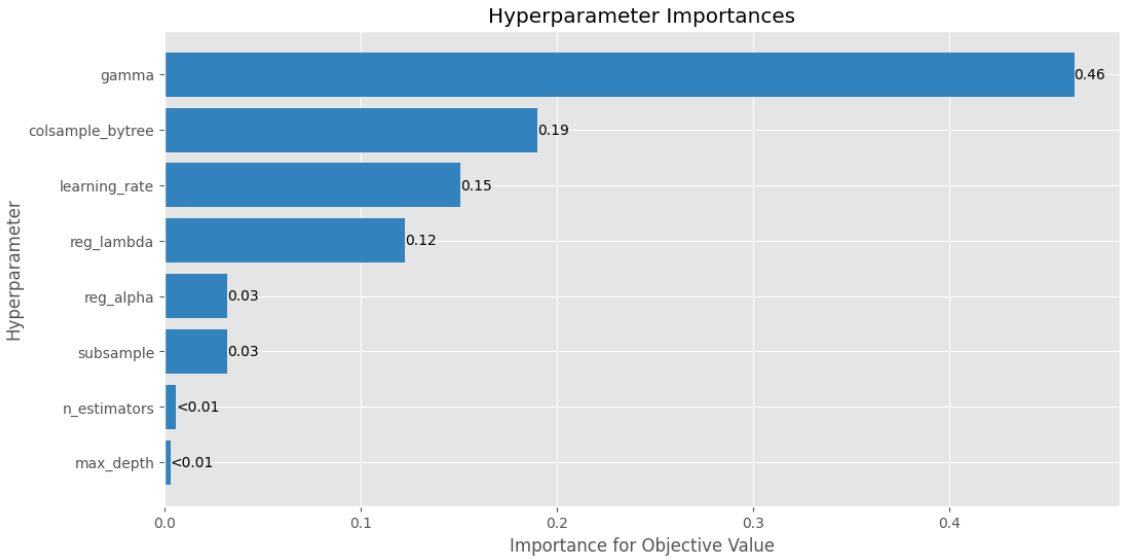


Figure 4.6: Importance of Hyperparameters for Objective Value

The selection of these hyperparameters aligns with the model documentation's recommendations, which suggest adjusting them in cases of overfitting [125].

In this methodology, the hyperparameters of SVR were also fine-tuned to enhance the model's performance. This process was completed using the Grid Search technique, an exhaustive search algorithm that considers all possible combinations of hyperparameters. Compared to the previously mentioned technique, Grid Search is more time-consuming and computationally intensive, especially when dealing with larger search spaces.

The selection of hyperparameters holds a pivotal role in this procedure. According to [126] and [127], the most critical hyperparameters for this model are regularization parameter (C) and gamma (γ).

Additionally, RBF kernel was preferred due to its flexibility in addressing various problems when compared to other kernel functions [128].

Table 4.3 summarizes the results of this procedure, which demonstrates slight improvements in the model's performance across the evaluated metrics.

Model	**MAE (ms)**		**R-Square**		**RMSE (ms)**	
	μ	σ	μ	σ	μ	σ
SVR Baseline	0.96923	0.13087	0.76338	0.08326	1.50154	0.23795
SVR Optimized	**0.94326**	0.11686	**0.78844**	0.07576	**1.48745**	0.21333

Table 4.3: Results of SVR Optimization Process.

4.3.2 Time Series Split

Unlike K-Fold Cross-Validation, Time Series Split method focuses on a more sequential and time-dependent framework. It leverages historical data to predict future observations, enabling the model to identify temporal dependencies and patterns within the data.

One key difference between these approaches resides in their data handling. While K-Fold Cross-Validation employs all folds simultaneously for training and testing, the one applied in this approach gradually increases the number of folds used for training as the time series data progresses, as shown in Figure 4.7.

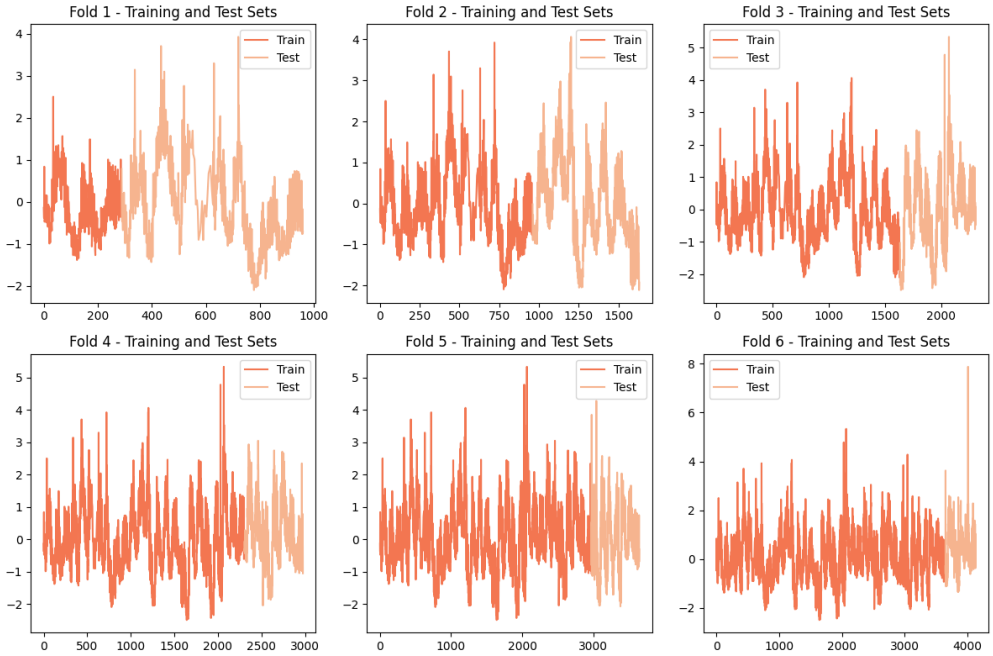


Figure 4.7: Train and Test Folds - Times Series Split

Like the previous study, various window sizes have been analyzed to estimate their impact on the model's effectiveness. Therefore, selecting an appropriate window size is a critical step toward achieving optimal performance in time series analysis.

As shown in Figure 4.8, smaller time windows may fail to capture essential data dependencies.

Despite their relatively lower overall performance, the models exhibiting superior performance were MLP and SVR. In turn, it highlights the model's challenges in learning and recognizing temporal data dependencies, hindering its generalization of unseen data.

At the same time, excessive information can lead to overfitting, diminishing the model's capacity to generalize and, consequently, resulting in worse performances.

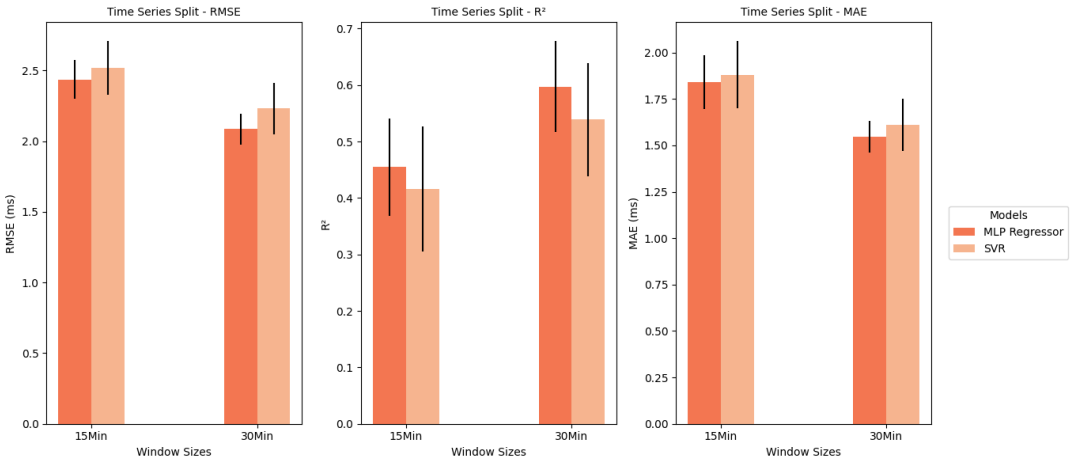


Figure 4.8: Comparison of Results between MLP and SVR for Smaller Window Sizes

One of the possible reasons for this behaviour is the introduction of more noise into the data, forcing the model to focus on the noise rather than learning the underlying patterns. Consequently, the model is tailored to the training data and performs poorly on unseen data.

Figure 4.9 illustrates this behaviour for one of the implemented models, namely MLP, whereas all the remaining ones follow the same direction.

These results underscore the importance of exploring multiple time windows when addressing time series problems.

Similar to the previous study, the models that conducted the best performance were MLP and SVR. Furthermore, the time windows with the most suitable results were consistent with the earlier approach, particularly at 45 minutes, 60 minutes and 90 minutes. These findings reinforce the consistency and reliability of the previous study's results.

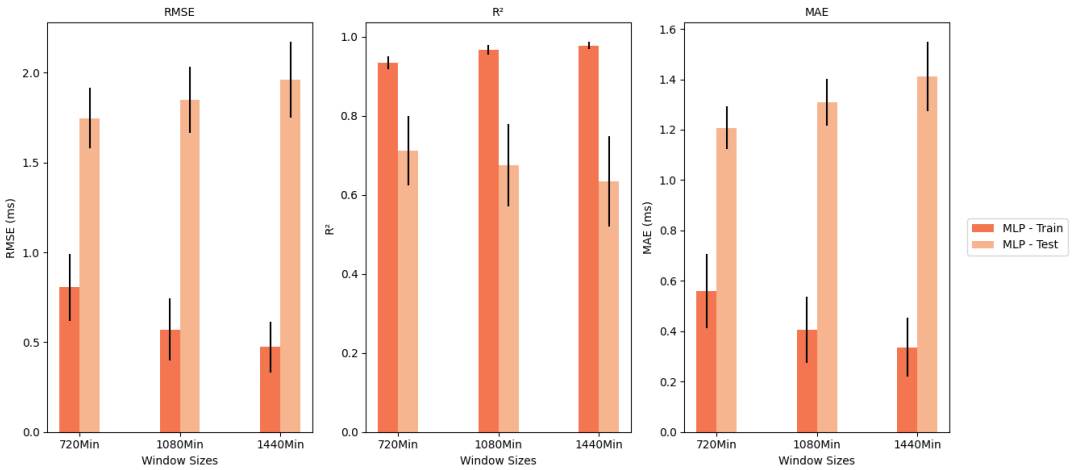


Figure 4.9: Comparison of Results between Larger Window Sizes for MLP

Figure 4.10 compares the performance of both models with regard to the optimal window sizes.

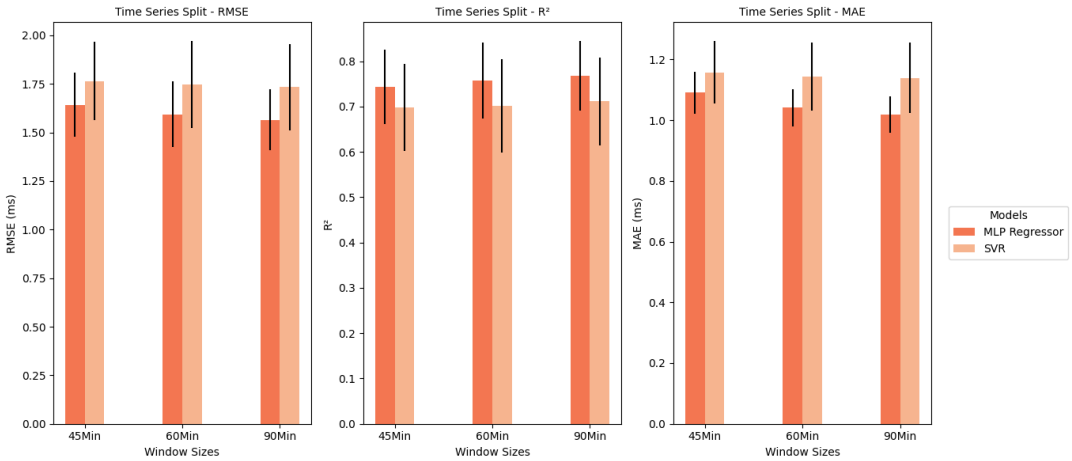


Figure 4.10: Result of Time Series Split Approach

In contrast to a previous study in which both models performed identically, the superior performance of MLP over SVR is evident when observing the figure. This reflects the efficacy of DNNs in accurately predicting network properties, as reported in the SoA literature.

Furthermore, Figures 4.11 and 4.12 display a comparison of results between the current approach and the previous one within the same time windows.

Figure 4.11 illustrates the performance of the MLP for both approaches. According to the results, the model produced similar outputs in both cases, with slightly higher results in K-Fold Cross-Validation.

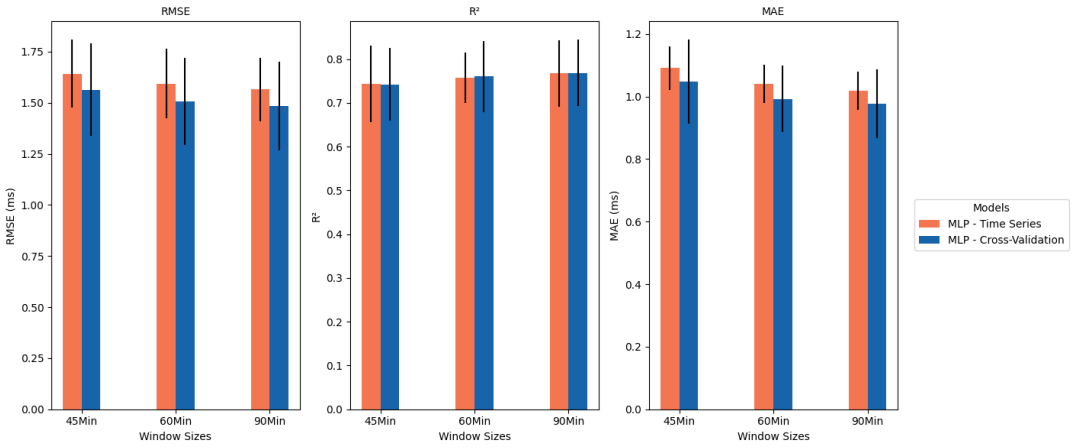


Figure 4.11: Comparison MLP Results between K-Fold Cross-Validation and Time Series Split

Regarding Figure 4.12, it presents the SVR model results, where evident differences emerge in the model's performance, especially in terms of RMSE and MAE values. Once more, the first technique, K-Fold Cross-Validation, exhibits superior performance.

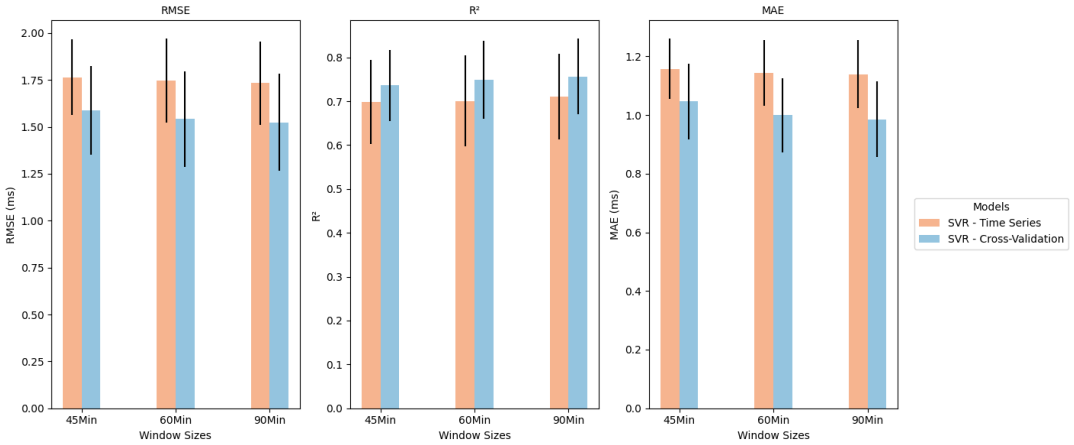


Figure 4.12: Comparison SVR Results between K-Fold Cross-Validation and Time Series Split

Building upon the previous methodology, the remaining models consistently show a tendency for overfitting. One significant cause for this behavior is the complexity of the models with the available data. However, another plausible factor is linked to data quality, a pivotal element for conducting realistic and dependable analyses.

Given that the data was gathered from an unstable and interference-prone network, it may contain substantial noise, posing a significant challenge for the models in accurately discerning underlying dependencies and achieving effective generalization to unobserved data. This noise comprises measurement errors, external interference or random system variations.

Due to the operating of the method, the presence of noise in the data within each fold limits the model's ability to consistently learn data connections.

One possible approach to address this issue is to augment the dataset by expanding its size. An alternative strategy, parallel to the preceding methodology, requires the optimization of the hyperparameters of the model. In this particular scenario, the employed methodology incorporates Grid Search and Bayesian Optimization approaches, as previously explained, but fails to provide the intended outcomes. Expressly, models with regularization techniques such as L1 and L2 incorporated into their architecture, like XGBoost, were given particular attention concerning these hyperparameters.

4.3.3 Train-Validation-Test Split

As stated in Section 2.3 relates to the SoA, the analyses conducted on articles similar to the purpose of this study conclude that the use of DNN architectures outperforms conventional ML algorithms in tasks related to predicting network attributes, such as latency.

Therefore, this experiment delves into integrating an LSTM architecture, one of the most widely used DNN architectures for handling time series data, to evaluate its performance in capturing data patterns associated with network latency. The analysis also examined the impact of various window sizes on the model's performance, as previously conducted in preceding methodologies.

Unlike the other experiments that employed techniques like Time Series Split and K-Fold Cross-Validation to evaluate model performance, this experiment employed a distinct methodology. In this case, the data were divided into 80% for training and the remaining 20% for validation and testing, as depicted in Figure 4.13. In particular, the training data consists of measurements that extend 30 days, while the validation and test data sets each represent seven days.

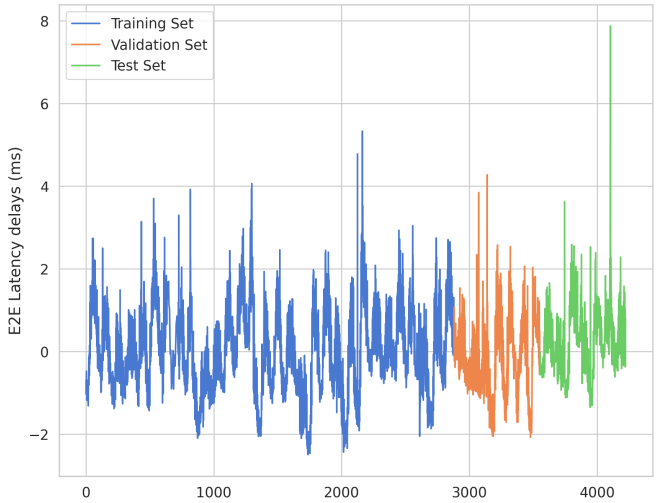


Figure 4.13: Dataset Division in Train-Validation-Test Approach

In this procedure, like remaining ones, the goal is to preserve as much temporal dependency as possible within the data and ensure that each generated subset contains the maximum number of completed weeks following the proposed ratio division. This procedure also empowers the model to recognize patterns and make more realistic inferences.

The LSTM was developed using the PyTorch framework [129]. In carrying out this methodology, a series of experiments were conducted to study and analyze the impact of several variables on the model's performance in order to identify the optimal hyperparameter configuration. These experiments involved modifications to the number of stacked layers, the number of neurons in the hidden layer, the learning rate and exploring various window sizes.

Since LSTM architectures rely on data-driven approaches and the limited available data for model training, optimizing the training process becomes crucial to prevent overfitting and enhance generalization performance. This necessity is further reinforced by the challenges faced in previous experiences with ML algorithms. As a result, two techniques were introduced to address this issue during the training process: integrating dropout layers into the model's architecture and implementing the early stopping method.

The first approach is a regularization technique by randomly deactivating a fraction of neurons during each training iteration [130]. Consequently, this procedure reduces the model's complexity, making the model less sensitive to overfitting. On the other hand, the early stopping method refers to monitoring the model's performance on a validation dataset during the training process. If the model's performance does not improve over a predefined number of epochs on a validation set, the model's training stops, preventing overfitting and saving computational resources. For this reason, more epochs, specifically 1000, were defined for the training process. Furthermore, the patient defined to stop the training process was ten epochs.

A learning rate schedule algorithm was also implemented in parallel with the last method. Among the available learning rate schedules, the `ReduceLROnPlateau` method was selected [131]. This method dynamically adjusts the learning rate during the model's training based on specific criteria. In turn, it allows the model to make more significant updates in the beginning when the loss is high and smaller updates as it converges, helping to find a global minimum of the objective function. In this case, it was defined as a patient of five epochs. If there are no improvements, the learning rate is reduced by a factor equal to 0.1.

Both methods work collaboratively; however, the training process stops if the predefined patience threshold is exceeded without improvement. If there is an improvement, the countdown restarts with the learning rate updated.

Moreover, two optimization algorithms were evaluated during the training process, especially Stochastic Gradient Descent (SGD) and ADAM. SGD is a ML optimization algorithm that addresses the computational inefficiency of traditional Gradient Descent [132] by using a single random training example instead of the entire dataset for each iteration.

This procedure introduces randomness into the optimization process, significantly reducing computational costs per iteration.

Additionally, ADAM consists of a popular variant of the SGD that incorporates adaptive learning rates for individual parameters, adjusting them based on the first and second moments of the gradients [133].

The objective was to determine the most suitable optimizer for updating the model's parameters to minimize the objective function.

Once the training process is completed, the weights of the models with the lowest loss for each performed experiment are saved in a dictionary, as illustrated in Figure 4.14. Subsequently, these models are validated on the test set.

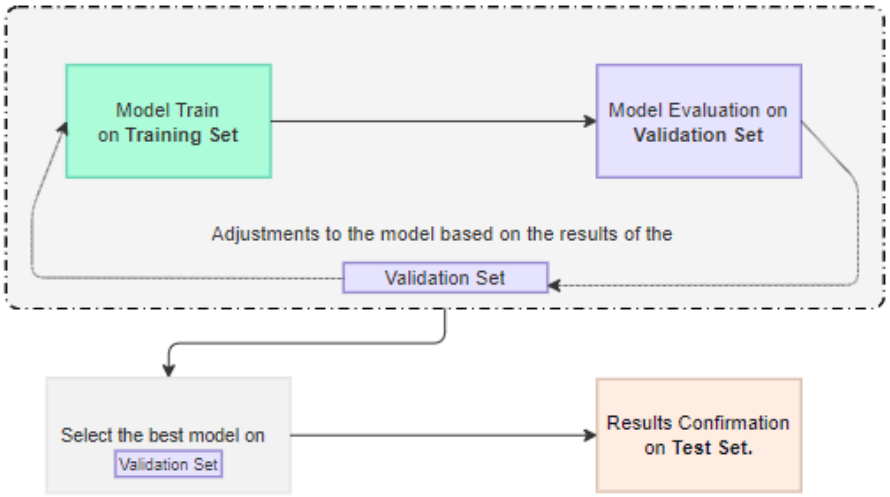


Figure 4.14: Illustration of the Train-Validation-Test Split procedure

The predicted values are rescaled to the normal values during the evaluation process to calculate the model's performance.

Compared to traditional ML algorithms evaluated in the previous section, one of the advantages of LSTM architecture is its ability to capture long-term data dependencies. This capacity is a result of the gates defined in their architecture, which control the flow of information within the network. It enables the model to discover data dependencies without losing information about inputs from the distant past. Contrarily, ML algorithms do not have this memory mechanism, assuming that each data point is independent from the others.

LSTM also manages the feature extraction phase autonomously, empowering it to capture more complex and abstract patterns.

In line with this, Table 4.4 exhibit baseline results for the time sequences that reached the best results.

Time Window	**MAE (ms)**	**R-Square**	**RMSE (ms)**
360 Min	1.46152	0.72402	0.89500
480 Min	1.62298	0.66877	0.92096
540 Min	1.60687	0.67493	1.01063
720 Min	1.72811	0.62395	1.01025
1080 Min	1.84721	0.58346	1.14955
1440 Min	1.83912	0.60185	1.106670

Table 4.4: Comparison of prediction performance of different time windows

The results reveal that time windows of 360 minutes, 540 minutes and 480 minutes are associated with superior outcomes. Among these, the first interval outperforms the others by a large margin. In this case, the average distance between the ground truth is less than one milliseconds, indicating promising results.

Additionally, the higher coefficient of relation reveals a greater capacity to explain the variation in future delays based on past values. Besides, more oversized time windows cause the worst results.

As mentioned, various configurations of hyperparameters were evaluated within the network architecture for each time window. These configurations included parameters such as the number of neurons in hidden layers, the number of stacked LSTM layers, the learning rate and the optimizer.

Table 4.5 summarizes the experiments with the most optimistic results, including information on the evaluated metrics and corresponding model hyperparameters.

Hyperparameters Configuration - LSTM Experiments										
ID	Time Window	Batch_Size	nº neurons	nº layers stacked	LR	Criterion	Optimizer	**MAE (ms)**	**R-Square**	**RMSE (ms)**
1	360Min	32	256	2	0.001	MSE	ADAM	1.35611	0.76239	0.86889
2	360Min	32	128	4	0.001	MSE	ADAM	1.40	0.74518	0.87024
3	360Min	32	128	5	0.001	MSE	ADAM	1.43088	0.73547	0.90634
4	360Min	32	128	3	0.001	MSE	ADAM	1.57402	0.6799	0.93080
5	540Min	32	128	4	0.001	MSE	ADAM	1.58400	0.68413	0.96558
6	720Min	16	256	2	0.001	MSE	ADAM	1.70228	0.63511	1.0166
7	480Min	32	256	2	0.001	MSE	ADAM	1.62825	0.6661	0.90088
8	1080Min	32	256	2	0.001	MSE	ADAM	1.82291	0.59435	1.14408
9	1440Min	32	128	3	0.001	MSE	ADAM	1.80496	0.6165	1.0437

Table 4.5: Hyperparameter Configurations and Results for LSTM Experiments

The results showcase that, for all time frames, when the model's complexity increases by introducing more neurons per layer or adding more LSTM layers, the model learns more intricate patterns and generates better results.

The experiments also demonstrated that the 360-minute time window significantly outperformed others, reaching the best results. ADAM was the most appropriate optimizer used during the experiment's evaluation. On the other hand, batch size 32 was the most suitable for the study.

Moreover, Figure 4.15 illustrates the best-performing experiment predictions, corresponding to ID 1 in Table 4.5.

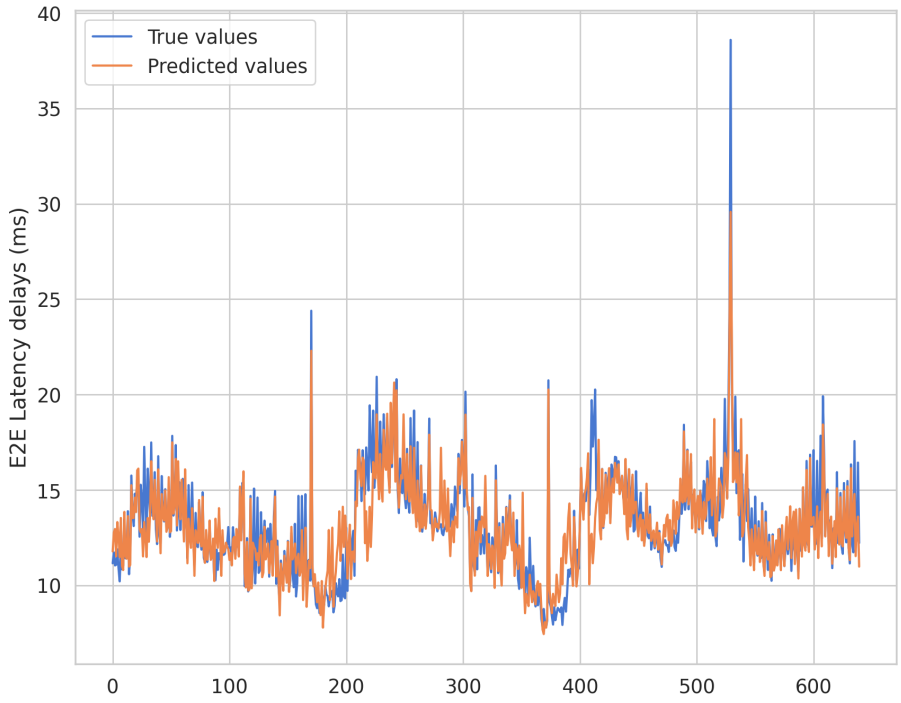


Figure 4.15: Network E2E latency Prediction using LSTM for Experiment ID 1

Lastly, the results from this methodology reinforce the superior adaptability of LSTM, as stated in the SoA, in predicting fluctuations in the E2E network latency over time compared to conventional machine learning algorithms.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Fifth Generation Mobile Networks and Beyond 5G are reshaping the parading of mobile networks, driving substantial progress in Vehicle-to-Everything (V2X) communications. These communications are critical components in connected and autonomous driving applications, requiring rigorous consistency with Quality of Service (QoS) performance standards. It supports uninterrupted and reliable data exchange among vehicles, infrastructure, and other entities present on the road. Additionally, integrating Artificial Intelligence with V2X communication unlocks new opportunities for intelligent traffic management and collaborative driving applications. All these elements together contribute to a more efficient and sustainable ecosystem of transport systems.

Within the scope of the work presented in this document, the proposed methodologies intended to provide valuable insight towards developing QoS prediction for V2X communication, with the aim of future contributions to intelligent transportation systems and intelligent driving applications, namely to Cooperative Intelligent Transportation Systems.

This research focuses on reliable network latency prediction in the complex environment of 5G networks to positively impact the safety, stability and efficiency of ITS. However, latency prediction is a challenging problem because various factors, such as network congestion, signal interference and unpredictable traffic patterns, influence it. Besides, the dynamic nature of V2X communication adds another layer of complexity to accurately predicting latency in real-time scenarios.

For this purpose, a range of ML algorithms were evaluated, including conventional ML algorithms and deep learning models. The goal was to determine the most accurate approach for latency prediction. Furthermore, the predictions considered various historical latency values and applied multiple techniques to comprehensively assess the model's performance. Overall, this extensive evaluation process aimed to enhance the reliability of the latency predictions, striving for the most realism and accuracy possible.

According to the results, LSTM performs better than traditional approaches implemented to enhance prediction accuracy.

This superior performance evidenced its adaptability to capture long-term dependencies and non-linear patterns in the data, given that its architecture acts like a ”memory” mechanism that retains information from previous past observations and learns to forget irrelevant information. However, the noise present in the data from various sources can challenge the accurate identification of these patterns, which impacts traditional ML models in terms of their performance and accuracy. Consequently, these models illustrated more difficulty learning a general pattern, enabling them to generalize unobserved data.

Lastly, the contributions of this work also extend to strategies for optimizing resource allocation within the network. In the context of the FLOYD project, the proposed architecture enables more efficient resource allocation and utilization through real-time data analysis, ultimately elevating the overall network performance.

5.2 Future Work

One of the difficulties identified during the project was linked to the quality of the data, which revealed inconsistencies. This inconsistency primarily arises from the inherent characteristics of the network, as extensively discussed in this document, being notably unstable. Furthermore, despite identifying and mitigating certain limitations in the testbed used for data collection during its development, it can still potentially contribute to data variance. Consequently, these interferences represent a significant challenge to the algorithm's capacity to discover patterns within the network.

In this context, improving the data quality becomes pivotal for validating the models and establishing more robust and reliable intelligent systems for latency prediction. This, in turn, drives towards a safer implementation of vehicles within the ITS domain.

As stated in the SoA, one limitation observed is also the absence of 5G data availability for conducting experiments and evaluating the algorithm's performance in real-world scenarios. As a result, one of the subsequent lines of investigation consists of improving the data quality by exploring data augmentation techniques.

At the same time, an additional future vision for the project is to collect other types of metrics, namely RAN metrics, as reported in a literature review. By incorporating these metrics, the aim is to construct models better suited to the diverse network characteristics. This approach will enable more comprehensive analyses, facilitating a deeper exploration of various factors that could potentially result in significant enhancements in both the understanding and performance of the models.

Moreover, other possible future research directions involve the development and integration of techniques for anomaly detection. While latency prediction focuses on forecasting delays and optimizing traffic flow, anomaly detection identifies unusual events and disruptions that could impact the system's operation.

The combination of these approaches has the potential to result in more efficient and resilient ITS solutions.

Ultimately, these limitations reinforce the importance of continuously studying and developing advanced algorithms and techniques to mitigate these factors and improve the accuracy of latency prediction. Therefore, this effort accelerates the integration of real-time data and predictive analytics into ITS systems, leading to more effective traffic management and improved overall transportation efficiency.

Bibliography

[1] Lei, Wan and Soong, Anthony and Jianghua, Liu and Yong, Wu and Classon, Brian and Xiao, Weimin and Mazzarese, David and Yang, Zhao and Saboorian, Tony, *5G System Design: An End to End Perspective*, 01 2020. 1, 10

[2] Refaee, Ali and Alshahrani, Abdullah and Muthanna, Ammar and Koucheryavy, Andrey, "Performance Estimation in V2X Networks Using Deep Learning-Based M-Estimator Loss Functions in the Presence of Outliers," *Symmetry*, vol. 13, p. 2207, 11 2021. 2

[3] FLOYD, "Floyd Project," https://www.cmuportugal.org/large-scale-collaborative-research-projects/floyd/. 2

[4] Joseph Thaliath and Solmaz Niknam and Sukhdeep Singh and Rahul Banerji and Navrati Saxena and Harpreet S. Dhillon and Jeffrey H. Reed and Ali Kashif Bashir and Avinash Bhat and Abhishek Roy, "Predictive Closed-Loop Service Automation in O-RAN based Network Slicing," 2022. 2

[5] Rashdan, Ibrahim and Sand, Stephan and de Ponte Müller, Fabian, "ITS-G5 Challenges and 5G Solutions for Vehicular Platooning," 01 2016. 5

[6] Dhinesh Kumar R and Rammohan A, "Revolutionizing Intelligent Transportation Systems with Cellular Vehicle-to-Everything (C-V2X) technology: Current trends, use cases, emerging technologies, standardization bodies, industry analytics and future directions," *Vehicular Communications*, vol. 43, p. 100638, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209623000682 5, 12

[7] Ferreira, Rui and Fonseca, João and Silva, João and Tendulkar, Mayuri and Duarte, Paulo and Araújo, Marco and Barbosa, Raul and Mendes, Bruno and Goes, Adriano, "Demo: Enhancing network performance based on 5g network function and slice load analysis," in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2023, pp. 340–342. 5, 55

[8] Panwar, Nisha and Sharma, Shantanu and Singh, Awadhesh, "A Survey on 5G: The Next Generation of Mobile Communication," *Physical Communication*, 01 2016. 7

[9] Niu, Yong and Li, Yong and Jin, Depeng and Su, Li and Vasilakos, Athanasios, "A Survey of Millimeter Wave (mmWave) Communications for 5G: Opportunities and Challenges," *Wireless Networks*, vol. 21, 02 2015. 7

[10] Vook, Frederick W. and Ghosh, Amitava and Thomas, Timothy A., "MIMO and beamforming solutions for 5G technology," in *2014 IEEE MTT-S International Microwave Symposium (IMS2014)*, 2014, pp. 1–4. 8

[11] Campolo, Claudia and Molinaro, Antonella and Iera, Antonio and Fontes, Ramon R. and Rothenberg, Christian E., "Towards 5G Network Slicing for the V2X Ecosystem," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 400–405. 8

[12] S. Wijethilaka and M. Liyanage, "Survey on network slicing for internet of things realization in 5g networks," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 957–994, 2021. 8

[13] Yu, Yifan, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges," *China Communications*, vol. 13, no. Supplement2, pp. 89–99, 2016. 8

[14] Duan, Qiang, "Intelligent and Autonomous Management in Cloud-Native Future Networks—A Survey on Related Standards from an Architectural Perspective," *Future Internet*, vol. 13, no. 2, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/2/42 8, 9

[15] Harper, Colby and Sirotkin, Sasha, *NG-RAN Architecture*. John Wiley & Sons, Ltd, 2021, ch. 4, pp. 123–234. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119550921.ch4 8, 13

[16] Cardoso, Kleber Vieira and Both, Cristiano Bonato and Prade, Lúcio Rene and Macedo, Ciro J. A. and Lopes, Victor Hugo L., "A softwarized perspective of the 5G networks," 2020. [Online]. Available: https://arxiv.org/abs/2006.10409 9, 10, 11, 12

[17] "Technical Specification Group Services and System Aspects; Release 16 Description; Summary of Rel-16 Work Items." 3rd Generation Partnership Project, Technical Report 21.916, ju 2021, accessed October 2023. [Online]. Available: https://www.3gpp.org/DynaReport/21916.htm 10

[18] 3rd Generation Partnership Project, "Technical Specification Group Services and System Aspects; Release 15 Description; Summary of Rel-15 Work Items ," 3rd Generation Partnership Project, Technical Report 21.915, sep 2019, accessed October 2023. [Online]. Available: https://www.3gpp.org/DynaReport/21915.htm 11

[19] Habibi, Mohammad Asif and Nasimi, Meysam and Han, Bin and Schotten, Hans D., "A Comprehensive Survey of RAN Architectures Toward 5G Mobile Communication System," *IEEE Access*, vol. 7, pp. 70 371–70 421, 2019. 12

[20] Gonçalves, Glauco and Leoni, Guto and Ferreira, Leylane and Rocha, Elisson and Souza, Lubnnia and Moreira, André and Kelner, Judith and Sadok, Djamel, *Flying to the Clouds: The Evolution of the 5G Radio Access Networks*, 07 2020, pp. 41–60. 13

[21] Sutton, Andy, *Enabling the Verticals of 5G: Network Architecture, Design and Service Optimization*. John Wiley & Sons, Ltd, 2019, ch. 1, pp. 1–21. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119515579.ch1 13

[22] Leonardo Bonati and Michele Polese and Salvatore D'Oro and Stefano Basagni and Tommaso Melodia, "Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead," *Computer Networks*, vol. 182, p. 107516, dec 2020. 13

[23] Sassan Ahmadi, "Chapter 1 - 5G Network Architecture," in *5G NR*. Academic Press, 2019, pp. 1–194. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780081022672000014 13

[24] "Technical Specification Group Services and System Aspects; Release 15 Description; Summary of Rel-15 Work Items ," 3rd Generation Partnership Project, Technical Report 21.915, sep 2019, accessed October 2023. [Online]. Available: https://www.3gpp.org/DynaReport/21915.htm 14

[25] International Telecommunication Union, "IMT Vision - Framework and overall objetives of the future development of IMT for 2020 and beyond." vol. M.2083-0, 2015. 14, 15, 16

[26] Mourtzis, Dimitris and Angelopoulos, John and Panopoulos, Nikos, "Smart Manufacturing and Tactile Internet Based on 5G in Industry 4.0: Challenges, Applications and New Trends," *Electronics*, vol. 10, no. 24, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/24/3175 15

[27] Navarro-Ortiz, Jorge and Romero-Diaz, Pablo and Sendra, Sandra and Ameigeiras, Pablo and Ramos-Munoz, Juan J. and Lopez-Soler, Juan M., "A Survey on 5G Usage Scenarios and Traffic Models," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 905–929, 2020. 14

[28] Islam, Md and Mahmud, Ashekraihan and Uddin, Md and Jodder, Tapan, "Analysis the Feasibility and Constraints of Implementing Next-Technology Mobile Networks in Bangladesh with its outcomes on Human Body," *International Journal of Data Science*, vol. 4, pp. 26–39, 05 2023. 14

[29] ETSI. (2022) 5g. [Online]. Available: https://www.etsi.org/technologies/mobile/5g 16

[30] Jiang, Dajie and Liu, Guangyi, *An Overview of 5G Requirements*. Springer International Publishing, 2017, pp. 3–26. [Online]. Available: https://doi.org/10.1007/978-3-319-34208-5_1 15

[31] Xiang, Wei and Zheng, Kan and Shen, Xuemin Sherman, *5G Mobile Communications*, 1st ed. Springer Publishing Company, Incorporated, 2016. 15

[32] 3rd Generation Partnership Project, “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Enhancement of 3GPP support for V2X scenarios; Stage 1 (Release 15) ,” 3rd Generation Partnership Project, Technical Report 22.186, march 2017, accessed November 2023. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3180 17

[33] ——, “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Enhancement of 3GPP support for V2X scenarios; Stage 1 (Release 15) ,” 3rd Generation Partnership Project, Technical Report 22.186, jun 2018, accessed November 2023. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3180 17, 18

[34] On-Road Automated Driving (ORAD) Committee, “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” sep 2016. [Online]. Available: https://doi.org/10.4271/J3016_201609 17

[35] Janiesch, Christian and Zschech, Patrick and Heinrich, Kathrin, “Machine learning and deep learning,” *Electron Markets*, vol. 31, pp. 685–695, 2021. 19, 20

[36] Morocho Cayamcela, Manuel Eugenio and Lim, Wansu, “Artificial Intelligence in 5G Technology: A Survey,” in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 860–865. 19

[37] Sayed, Sayed and Abdel Hamid, Yasser and Hefny, Hesham, “Artificial Intelligence based Traffic Flow Prediction a Comprehensive Review,” *Journal of Electrical Systems and Information Technology*, vol. 10, pp. 1–42, 03 2023. 19, 27, 29, 49

[38] P. Cunningham, M. Cord, and S. J. Delany, *Supervised Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49. [Online]. Available: https://doi.org/10.1007/978-3-540-75171-7_2 20

[39] Fourati, Hasna and Maaloul, Rihab and Fourati, Lamia, “A survey of 5G network systems: challenges and machine learning approaches,” *International Journal of Machine Learning and Cybernetics*, vol. 12, 2021. 20, 23, 29

[40] Yu, F. Richard and Yu, Angela W., *Intelligence in Machines*. Springer International Publishing, 2023, pp. 65–88. [Online]. Available: https://doi.org/10.1007/978-3-031-15951-0_7 20

[41] Jin, Xin and Han, Jiawei, *K-Means Clustering*. Boston, MA: Springer US, 2010, pp. 563–564. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_425 20

[42] S. Velliangiri and S. Alagumuthukrishnan and S Iwin Thankumar joseph, "A Review of Dimensionality Reduction Techniques for Efficient Computation," *Procedia Computer Science*, vol. 165, pp. 104–111, 2019, 2nd International Conference on Recent Trends in Advanced Computing ICRTAC -DISRUP - TIV INNOVATION , 2019 November 11-12, 2019. 20

[43] Zebari, Rizgar and Mohsin Abdulazeez, Adnan and Zeebaree, Diyar and Zebari, Dilovan and Saeed, Jwan, "A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction," *Journal of Applied Science and Technology Trends*, vol. 1, pp. 56–70, 05 2020. 20

[44] Toshniwal, Akanksha and Mahesh, Kavi and Jayashree, R., "Overview of Anomaly Detection techniques in Machine Learning," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2020, pp. 808–815. 20

[45] Reddy, Y and Pulabaigari, Viswanath and B, Eswara, "Semi-supervised learning: a brief review," *International Journal of Engineering & Technology*, vol. 7, p. 81, 02 2018. 21

[46] Sutton, R.S. and Barto, A.G., "Reinforcement Learning: An Introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1054–1054, 1998. 21

[47] L. P. Kaelbling and M. L. Littman and A. W. Moore, "Reinforcement Learning: A Survey," 1996. 21

[48] Hinton, Geoffrey and Osindero, Simon and Teh, Yee-Whye, "A Fast Learning Algorithm for Deep Belief Nets," *Neural computation*, vol. 18, pp. 1527–54, 08 2006. 21

[49] Laith Alzubaidi and Jinglan Zhang and Amjad J. Humaidi and Ayad Al-Dujaili and Ye Duan and Omran Al-Shamma and J. Santamaría and Mohammed A. Fadhel and Muthana Al-Amidie and Laith Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, p. Article number: 53, December 2021. 21, 22

[50] Shunliang Zhang and Dali Zhu, "Towards artificial intelligence enabled 6G: State of the art, challenges, and opportunities," *Computer Networks*, vol. 183, p. 107556, 2020. 21

[51] Zhang, Chaoyun and Patras, Paul and Haddadi, Hamed, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019. 21, 22, 23, 29, 52

[52] Keiron O'Shea and Ryan Nash, "An Introduction to Convolutional Neural Networks," 2015. 22

[53] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361. 22

[54] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E., "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105. 22

[55] Christian Szegedy and Wei Liu and Yangqing Jia and Pierre Sermanet and Scott Reed and Dragomir Anguelov and Dumitru Erhan and Vincent Vanhoucke and Andrew Rabinovich, "Going Deeper with Convolutions," 2014. 22

[56] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition," 2015. 22

[57] Robin M. Schmidt, "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview," 2019. 22

[58] Sarker, Iqbal, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Computer Science*, vol. 2, 08 2021. 22, 51

[59] Hochreiter, Sepp and Schmidhuber, Jürgen, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997. 22, 27, 51

[60] Kyunghyun Cho and Bart van Merrienboer and Caglar Gulcehre and Dzmitry Bahdanau and Fethi Bougares and Holger Schwenk and Yoshua Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," 2014. 22

[61] Pascanu, Razvan and Mikolov, Tomas and Bengio, Y., "On the difficulty of training Recurrent Neural Networks," *30th International Conference on Machine Learning, ICML 2013*, 11 2012. 22

[62] Zhao, Yanling and Li, Ye and Zhang, Xinchang and Geng, Guanggang and Zhang, Wei and Sun, Yanjie, "A Survey of Networking Applications Applying the Software Defined Networking Concept Based on Machine Learning," *IEEE Access*, vol. 7, pp. 95 397–95 417, 2019. 23, 29, 47

[63] B. Wei, W. Kawakami, K. Kanai, and J. Katto, "A history-based tcp throughput prediction incorporating communication quality features by support vector regression for mobile network," in *2017 IEEE International Symposium on Multimedia (ISM)*, 2017, pp. 374–375. 23, 24, 29

[64] Muhammad Asif Khan and Ridha Hamila and Nasser Ahmed Al-Emadi and Serkan Kiranyaz and Moncef Gabbouj, "Real-time throughput prediction for cognitive Wi-Fi networks," *Journal of Network and Computer Applications*, vol. 150, p. 102499, 2020. 23, 29

[65] Zhang, Wenhan and Feng, Mingjie and Krunz, Marwan and Volos, Haris, "Latency Prediction for Delay-sensitive V2X Applications in Mobile Cloud/Edge Computing Systems," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6. 24, 29

[66] M. Samiuddin and G. M. El-Sayyad, "On Nonparametric Kernel Density Estimates," *Biometrika*, vol. 77, no. 4, pp. 865–874, 1990. [Online]. Available: http://www.jstor.org/stable/2337109 24

[67] Hyeonjun Na and Yongjoo Shin and Dongwon Lee and Joohyun Lee, "LSTM based throughput prediction for LTE networks," *ICT Express*, vol. 9, no. 2, pp. 247–252, 2023. 24, 29, 51

[68] Sutskever, Ilya and Vinyals, Oriol and Le, Quoc V, "Sequence to Sequence Learning with Neural Networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf 24

[69] Alexander M. Rush and Sumit Chopra and Jason Weston, "A Neural Attention Model for Abstractive Sentence Summarization," 2015. 24

[70] Minovski, Dimitar and Ögren, Niclas and Mitra, Karan and Åhlund, Christer, "Throughput Prediction Using Machine Learning in LTE and 5G Networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1825–1840, 2023. 25, 29, 59

[71] Schmid, Josef and Schneider, Mathias and Hoess, Alfred and Schuller, Björn, "A Deep Learning Approach for Location Independent Throughput Prediction," 11 2019, pp. 1–5. 25, 29

[72] Moreira, Darlan C. and Guerreiro, Igor M. and Sun, Wanlu and Cavalcante, Charles C. and Sousa, Diego A., "QoS Predictability in V2X Communication with Machine Learning," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5. 25, 29

[73] S.L. Ho and M. Xie, "The use of ARIMA models for reliability forecasting and analysis," *Computers & Industrial Engineering*, vol. 35, no. 1, pp. 213–216, 1998. 25

[74] Külzer, Daniel Fabian and Debbichi, Firas and Stańczak, Sławomir and Botsov, Mladen, "On Latency Prediction with Deep Learning and Passive Probing at High Mobility," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–7. 25, 29, 47

[75] Abdellah, Ali R. and Muthanna, Ammar and Essai, Mohamed H. and Koucheryavy, Andrey, "Deep Learning for Predicting Traffic in V2X Networks," *Applied Sciences*,

vol. 12, no. 19, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/19/10030 25, 29

[76] Marcus Kalander and Min Zhou and Chengzhi Zhang and Hanling Yi and Lujia Pan, "Spatio-Temporal Hybrid Graph Convolutional Network for Traffic Forecasting in Telecommunication Networks," 2020. 26, 29, 33, 36

[77] Trinh, Hoang Duy and Giupponi, L. and Dini, Paolo, "Mobile Traffic Prediction from Raw Data Using LSTM Networks," 09 2018. 26, 29

[78] Gao, Zihang and Gu, Yang, "5G Traffic Prediction Based on Deep Learning," *Intell. Neuroscience*, vol. 2022, jan 2022. [Online]. Available: https://doi.org/10.1155/2022/3174530 26, 29

[79] Marco Skocaj and Francesca Conserva and Nicol Sarcone Grande and Andrea Orsi and Davide Micheli and Giorgio Ghinamo and Simone Bizzarri and Roberto Verdone, "Data-driven Predictive Latency for 5G: A Theoretical and Experimental Analysis Using Network Measurements," 2023. 26, 33

[80] Yaqi Xu and Yan Shi and Yuming Ge and Shanzhi Chen and Longxiang Wang, "Informer-based QoS prediction for V2X communication: A Method With Verification Using Reality Field Test Data," *Computer Networks*, vol. 235, p. 109958, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623004036 27, 29, 32

[81] Xu, Lingwei and Wang, Han and Gulliver, T. Aaron, "Outage Probability Performance Analysis and Prediction for Mobile IoV Networks Based on ICS-BP Neural Network," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3524–3533, 2021. 27

[82] j. Xu, Lingwei and Zhou, Xinpeng and Khan, Mohammad Ayoub and Li, Xingwang and Menon, Varun G and Yu, Xu, "Communication Quality Prediction for Internet of Vehicle (IoV) Networks: An Elman Approach," vol. 23, no. 10, pp. 19 644–19 654, 2022. 27

[83] C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5g wireless communications: A deep learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 227–236, 2020. 27

[84] Akem, Aristide T.-J. and Mugume, Edwin, "A Machine Learning Approach to Temporal Traffic-Aware Energy-Efficient Cellular Networks," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2020, pp. 0628–0634. 27, 29, 37, 53

[85] Komorowski, Matthieu and Marshall, Dominic and Salciccioli, Justin and Crutain, Yves, *Exploratory Data Analysis*, 09 2016, pp. 185–203. 31

[86] Hoitash, Rani and Kogan, Alexander and Srivastava, Rajendra and Vasarhelyi, Miklos, "Measuring Information Latency," *International Journal of Digital Accounting Research*, 01 2006. 32

[87] Fan, Cheng and Chen, Meiling and Wang, Xinghua and Wang, Jiayuan and Huang, Bufu, "A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data," *Frontiers in Energy Research*, vol. 9, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fenrg.2021.652801 33

[88] Aleksey Bilogur, "Missingno: a missing data visualization suite," *Journal of Open Source Software*, vol. 3, no. 22, p. 547, 2018. [Online]. Available: https://doi.org/10.21105/joss.00547 34

[89] Lepot, Mathieu and Aubin, Jean-Baptiste and Clemens, François H.L.R., "Interpolation in Time Series: An Introductive Overview of Existing Methods, Their Performance Criteria and Uncertainty Assessment," *Water*, vol. 9, no. 10, 2017. [Online]. Available: https://www.mdpi.com/2073-4441/9/10/796 34

[90] Pandas development team. (2023) pandas.dataframe.interpolate. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html 35

[91] M. Kulesh and M. Holschneider and K. Kurennaya, "Adaptive metrics in the nearest neighbours method," *Physica D: Nonlinear Phenomena*, vol. 237, no. 3, pp. 283–291, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167278907003338 35

[92] Shichao Zhang and Zhi Jin and Xiaofeng Zhu, "Missing data imputation by utilizing information within incomplete instances," *Journal of Systems and Software*, vol. 84, no. 3, pp. 452–459, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121210003092 35

[93] de Amorim, Lucas and Cavalcanti, George and Cruz, Rafael, "The choice of scaling technique matters for classification performance," *Appl. Soft Comput.*, vol. 133, p. 109924, 12 2022. 36

[94] Sadiku, Matthew and Shadare, Adebowale and Musa, Sarhan and Akujuobi, Cajetan and Perry, Roy, "DATA VISUALIZATION," *International Journal of Engineering Research and Advanced Technology (IJERAT)*, vol. 12, pp. 2454–6135, 12 2016. 36

[95] Williamson, D and Parker, RA and Kendrick, Juliette, "The box plot: A simple visual method to interpret data," *Annals of internal medicine*, vol. 110, pp. 916–21, 07 1989. 37

[96] Shivam Chaudhary. (2019) Why “1.5” in IQR Method of Outlier Detection? [Online]. Available: https://towardsdatascience.com/why-1-5-in-iqr-method-of-outlier-detection-5d07fdc82097 38

[97] Scikit-Learn. Scikit-Learn - Density Estimation. [Online]. Available: https://scikit-learn.org/stable/modules/density.html 41

[98] Michael L. Waskom, “Seaborn: Statistical Data Visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021 42

[99] Moreno-Torres, Jose García and Saez, José A. and Herrera, Francisco, “Study on the Impact of Partition-Induced Dataset Shift on k-Fold Cross-Validation,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1304–1312, 2012. 44

[100] Refaeilzadeh, Payam and Tang, Lei and Liu, Huan, *Cross-Validation*. Boston, MA: Springer US, 2009. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_565 44

[101] Sylvain Arlot and Alain Celisse, “A survey of Cross-Validation Procedures for Model Selection,” *Statistics Surveys*, vol. 4, no. none, pp. 40 – 79, 2010. [Online]. Available: https://doi.org/10.1214/09-SS054 45

[102] Scikit-Learn. Scikit-Learn - Machine Learning in Python. [Online]. Available: https://scikit-learn.org/stable/ 47

[103] Chen, Tianqi and Guestrin, Carlos, “XGBoost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785 47

[104] ——, “XGBoost: A Scalable Tree Boosting System,” 08 2016, pp. 785–794. 47

[105] Ke, Guolin and Meng, Qi and Finley, Thomas and Wang, Taifeng and Chen, Wei and Ma, Weidong and Ye, Qiwei and Liu, Tie-Yan, “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. 48

[106] Alzamzami, Fatimah and Hoda, Mohamad and Saddik, Abdulmotaleb El, “Light Gradient Boosting Machine for General Sentiment Classification on Short Texts: A Comparative Evaluation,” *IEEE Access*, vol. 8, pp. 101 840–101 858, 2020. 48

[107] Breiman, L, “Random Forests,” *Machine Learning*, vol. 45, pp. 5–32, 10 2001. 49

[108] Ali, Jehad and Khan, Rehanullah and Ahmad, Nasir and Maqsood, Imran, "Random Forests and Decision Trees," *International Journal of Computer Science Issues(IJCSI)*, vol. 9, 09 2012. 49

[109] Vapnik, Vladimir and Golowich, Steven E. and Smola, Alex, "Support Vector Method for Function Approximation, Regression Estimation and Signal Processing," ser. NIPS'96. Cambridge, MA, USA: MIT Press, 1996, p. 281–287. 50

[110] Ma, Yongchang and Chowdhury, Mashrur and Sadek, Adel and Jeihani, Mansoureh, "Integrated Traffic and Communication Performance Evaluation of an Intelligent Vehicle Infrastructure Integration (VII) System for Online Travel-Time Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1369–1382, 2012. 50

[111] Abbas, Farhat and Afzaal, Hassan and Farooque, Aitazaz A. and Tang, Skylar, "Crop Yield Prediction through Proximal Sensing and Machine Learning Algorithms," *Agronomy*, vol. 10, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2073-4395/10/7/1046 50

[112] Singh, Abhilash and Kotiyal, Vaibhav and Sharma, Sandeep and Nagar, Jaiprakash and Lee, Cheng-Chi, "A Machine Learning Approach to Predict the Average Localization Error With Applications to Wireless Sensor Networks," *IEEE Access*, vol. 8, pp. 208 253 – 208 263, 11 2020. 50

[113] Dubey, Shiv Ram and Singh, Satish Kumar and Chaudhuri, Bidyut, "Activation Functions in Deep Learning: A comprehensive Survey and Benchmark," *Neurocomputing*, vol. 503, 07 2022. 52

[114] Hecht-Nielsen, "Theory of the backpropagation neural network," in *International 1989 Joint Conference on Neural Networks*, 1989, pp. 593–605 vol.1. 52

[115] Sarraf Shirazi, Alireza and Frigaard, Ian, "SlurryNet: Predicting Critical Velocities and Frictional Pressure Drops in Oilfield Suspension Flows," *Energies*, vol. 14, p. 1263, 02 2021. 52

[116] Chicco, Davide and Warrens, Matthijs and Jurman, Giuseppe, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, 07 2021. 53

[117] Ratnadip Adhikari and R. K. Agrawal, "An Introductory Study on Time Series Modeling and Forecasting," 2013. 54

[118] Vitor Cerqueira and Luis Torgo and Igor Mozeti, "Evaluating time series forecasting models: an empirical study on performance estimation methods," *Machine Learning*, vol. 109, no. 11, pp. 1997–2028, oct 2020. 56

[119] Bergstra, James and Bengio, Yoshua, “Random Search for Hyper-Parameter Optimization,” *J. Mach. Learn. Res.*, vol. 13, p. 281–305, feb 2012. 59

[120] Akiba, Takuya and Sano, Shotaro and Yanase, Toshihiko and Ohta, Takeru and Koyama, Masanori, “Optuna: A Next-Generation Hyperparameter Optimization Framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2019, p. 2623–2631. 60

[121] Bergstra, James and Bardenet, Remi and Bengio, Yoshua and Kégl, Balazs, “Algorithms for hyper-parameter optimization,” in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. 60

[122] O. Community. (2023) Optuna documentation - tpesampler. Accessed March 2023. [Online]. Available: https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.TPESampler.html 60

[123] Optuna Community. (2023) Optuna Documentation - Efficient Optimization Algorithms. Accessed on March 2023. 61

[124] Lisha Li and Kevin Jamieson and Giulia DeSalvo and Afshin Rostamizadeh and Ameet Talwalkar, “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization.” 61

[125] XGBoost Contributors. (2023) XGBoost Documentation - Notes on Parameter Tuning. Accessed on March 2023. [Online]. Available: https://xgboost.readthedocs.io/en/stable/tutorials/param_tuning.html 62

[126] Mantovani, Rafael G. and Rossi, André L. D. and Vanschoren, Joaquin and Bischl, Bernd and de Carvalho, André C. P. L. F., “Effectiveness of Random Search in SVM hyper-parameter tuning,” in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8. 63

[127] Alrefaee, Saifuldeen and Al Bakal, Salih and Algamal, Zakariya, “Hyperparameters optimization of support vector regression using black hole algorithm,” *The International Journal of Nonlinear Analysis and Applications (IJNAA)*, vol. 13, pp. 3441–3450, 01 2022. 63

[128] Gomes, Taciana A. F. and Prud□ncio, Ricardo B. C. and Soares, Carlos and Rossi, Andre L. D. and Carvalho, Andre, “Combining Meta-learning and Search Techniques to SVM Parameter Selection,” in *2010 Eleventh Brazilian Symposium on Neural Networks*, 2010, pp. 79–84. 63

[129] Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin,

Zeming and Gimelshein, Natalia and Antiga, Luca and Desmaison, Alban and Kopf, Andreas and Yang, Edward and DeVito, Zachary and Raison, Martin and Tejani, Alykhan and Chilamkurthy, Sasank and Steiner, Benoit and Fang, Lu and Bai, Junjie and Chintala, Soumith, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf 68

[130] Srivastava, Nitish and Hinton, Geoffrey and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929–1958, jan 2014. 68

[131] P. Contributors. (2023) PyTorch Documentation: ReduceLROnPlateau. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html 68

[132] Ruder, Sebastian, "An overview of gradient descent optimization algorithms," 09 2016. 68

[133] Kingma, Diederik and Ba, Jimmy, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 12 2014. 69