Deliverable D5.7

Framework for Cognitive SLA and QoE Slice Management

Editors:	Kenneth Nagin, IBM Research Haifa (IBM) Salvatore Spadaro, Universitat Politècnica de Catalunya (UPC)
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	30/11/19
Actual delivery date:	08/12/19
Suggested readers:	Network Administrators, Vertical Industries, Telecommunication Vendors, Telecommunication Operators, Service Providers
Version:	1.0
Total number of pages:	70
Keywords:	Network Slicing, AIOPS 5G, Cognitive management, MAPE, noisy neighbours, RAN optimisation, reliable RAN

Abstract

This document reports the third iteration of the SliceNet Cognition Sub-Plane design and implementation. SliceNet Cognition Sub-Plane enables the Quality of Experience (QoE)-aware management of network slices. SliceNet QoE-aware slice management combines the established MAPE (Monitoring, Analysis, Planning, and Execution) autonomic control loop with state-of-the-art data-driven management and AIOPS (Artificial Intelligence for IT Operations). To this end, the components reported in this document provide the implementations of both analytical and actuation frameworks of the Cognition Sub-Plane, all governed through a data-centric approach. The current document also describes the role played by the Cognitive Sub-Plane in the SliceNet Use Cases, progress to the previous iteration's analytic workflows, the position of SliceNet's Cognition Sub-Plane against related 5G standards and other research projects. It also discusses how the Cognition Sub-Plane has achieved its goals and how it can be exploited beyond the scope of SliceNet.

Disclaimer

The research leading to these results has received funding from the European Union Horizon 2020 Programme under grant agreement number 761913.

Impressum

[Full project title]	End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualized Multi-Domain, Multi-Tenant 5G Networks
[Short project title]	SliceNet
[Number and title of work-package]	WP5 - Cognitive, Service-Level QoE Management
[Number and title of tasks]	T5.1 Framework for Cognitive SLA and QoE Slice Management; T5.3. Modelling, Design and Implementation of QoE Monitoring, Analytics and Optimisation Engine; T5.4 Modelling, Design and Implementation of Vertical-Informed QoE Actuators
[Document title]	Framework for Cognitive SLA and QoE Slice Management
[Editor: Name, company]	Kenneth Nagin, IBM Research - Haifa (IBM); Salvatore Spadaro, Universitat Politècnica de Catalunya (UPC)
[Work-package leader]	Kenneth Nagin, IBM Research - Haifa (IBM)

Copyright notice

Executive summary

The provisioning of network slices (NSes) with proper Quality of Experience (QoE) guarantees is seen as one of the key enablers of future 5G networks. However, it poses several challenges in the slices management that need to be addressed for efficient end-to-end (E2E) services delivery, including estimating QoE Key Performance Indicators (KPIs) from monitored metrics and reconfiguration operations (actuations) to support and maintain the desired quality levels. SliceNet provides a design and implementation of cognitive slice management that leverages Machine Learning (ML) techniques to proactively maintain the network in the required state to assure E2E QoE, as perceived by the vertical customers.

This deliverable is the third iteration of the overall Cognition Sub-Plane design and implementation. It demonstrates work package-level integration of the different Cognition Sub-Plane components and initial integration with other SliceNet components (such as Monitoring, FCAPS (Fault, Configuration, Accounting, Performance and Security) management, and Plug and Play (P&P) control) and applies this to the three SliceNet vertical Use Cases (UCs), namely 5G Smart Grid Self-Healing, e-Health connected ambulance, and Smart City. These UCs provide a means to validate SliceNet's approach and architecture for QoE cognitive management of NSes, including interfaces and prototypes. In addition to the UC integration, the deliverable describes progress on the analytic workflows introduced in the previous two iterations. The deliverable also describes the alignment of the final SliceNet's Cognition Sub-Plane with related 5G standards and how the Cognitive Sub-Plane compliments other related 5G Projects with additional capabilities. Finally, it shows how the Cognition Sub-Plane works with the rest of SliceNet architecture components to assure NS E2E QoE.

List of authors

Company	Author
Altice Labs SA, Portugal	Pedro Neves , Rui Pedro, Guilherme Cardoso, Nuno Henriques
Eurecom	Navid Nikaein, Nasim Ferdosian, Berkay Koksal
IBM Research – Haifa	Kenneth Nagin
Orange Romania	Marius Iordache, Catalin Brezeanu
Orange SA, France	Yosra Ben Slimen, Joanna Balcerzak, Imen Grida Ben Yahia
Universitat Politècnica de Catalunya	Albert Pagès, Fernando Agraz, Salvatore Spadaro, Rafael Montero

List of reviewers

Company	Reviewer
University of the West of Scotland (UWS)	Qi Wang
Orange SA, France	Joanna Balcerzak

Table of contents

List of figures

List of tables

Abbreviations

5G	Fifth Generation (mobile/cellular networks)
5G PPP	5G Infrastructure Public Private Partnership
AIOPS	Artificial Intelligence for IT Operations
AMC	Autonomic Management and Control
API	Application Programming Interface
AuN	Autonomic Networking
bDDN	Big-Data-Driven Networking
CESC	Cloud Enabled Small Cell
CESCM	CESC Manager
CPU	Central Processing Unit
CQI	Channel Quality Indicator
CSV	Comma Separated Values
DC	Data Centre
DE	Decision Entity
DSP	Digital Service Provider
E2E	End-to-end
ECA	Event-Condition-Action
EMS	Element Management System
eNB	Evolved Node B
ENI	Experiential Networked Intelligence
FCAPS	Fault, Configuration, Accounting, Performance and Security
FN	False Negative
FP	False Positive
GANA	Generic Autonomic Network Architecture
IED	Intelligent Electronic Device
IoT	Internet of Things
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
ISG	Industry Specification Group
JSON	Java Script Notation Object
KB	Knowledge Base
KPI	Key Performance Indicator
MAPE-K	Monitoring, Analysis, Planning and Execution governed by a Knowledge-base
ML	Machine Learning
MLP	Multi-Layer Perceptron
MOS	Mean Opinion Score
MQTT	Message Queuing Telemetry Transport
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NN	Noisy Neighbour
NO	Network Operator
NS	Network Slice
NSP	Network Service Provider
NSS	Network Sub-Slice
OAI	Open Air Interface
OSA	One Stop API
P&P	Plug & Play

PAP	Policy Administration Point
PDP	Policy Decision Point
PF	Policy Framework
PM	Physical Machine
PNF	Physical Network Function
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RCA	Root Cause Analysis
RRH	Remote Radio Head
RTT	Round Trip Time
SDN	Software Defined Networking
SG	Smart Grid
SIM	Subscriber Identification Module
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SliceNet	End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualized Multi-Domain, Multi-Tenant 5G Networks
SON	Self Organizing Network
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TP	True Positive
TUC	Technical Use Case
UC	Use Case
UDP	User Datagram Protocol
UE	User Equipment
ULL	Ultra-Low Latency
VM	Virtual Machine
VNF	Virtual Network Function
VoIP	Voice over IP
wbCQI	Wide-band CQI
WP	Work Package

1 Introduction

Across three iterations, SliceNet has built and prototyped a Cognition Sub-Plane architecture with the aim to provide the tools for achieving QoE-aware management of NSes. This has been accomplished thanks to the implementation of a full Monitoring, Analysis, Planning and Execution governed by a Knowledge-base (MAPE-K) loop that encloses QoE monitoring and analysis functions as well as an actuation system to apply remedies based on the monitoring/analysis outputs for QoE maintenance. Moreover, the developed Cognition Sub-Plane engages with other modules of the SliceNet architecture to achieve its purposes; namely, the FCAPS management system developed in WP6 for obtaining data inputs, essential for QoE monitoring/analysis, and the Orchestration system developed in WP7 as the entry point to enforce desired actuations to guarantee the QoE of the multiple slices.

In this last iteration, we focus on how the Cognitive Sub-Plane and QoE Slice Management apply to the overall architecture and how it is specifically relevant to the three vertical UCs defined in SliceNet. We also discuss enhancement to the analytic workflows presented in the previous two iterations that explore how to best satisfy the E2E QoE by exploiting ML techniques.

Finally, to give closure to the work developed, we review the position of SliceNet's Cognition Sub-Plane against related 5G standards and other 5G research projects as well as discussing how the Cognition Sub-Plane has achieved its goals, lessoned learned and how it can be exploited beyond the scope of SliceNet.

1.1 Document structure

This document is structured as follows:

1. Section 2 describes the role played by the Cognitive Sub-Plane in each of the SliceNet UCs.
2. Section 3 describes progress to the analytic workflows presented in the previous iteration of WP5, namely D5.6.
3. Section 4 describes Cognitive Sub-Plane's alignment with related 5G Standards.
4. Section 5 describes Cognitive Sub-Plane's compatibility with other 5G Projects and its innovations beyond the said projects.
5. Section 6 describes the essential contributions of the Cognitive Sub-Plane to the SliceNet Architecture and why it matters.
6. Lastly, Section 7 concludes with lessons learned and future directions.

2 Cognitive QoE Slice Management Per Use Case

2.1 Overview

This section discusses how the Cognitive Sub-Plane (described in detail in the previous iterations, that is, D5.5 [1] and D5.6 [2]) is instantiated to support the SliceNet vertical UCs. The general details about the UCs prototyping can be found in D7.2 [3]. As such, in this document, we focus on the role of the developed Cognition Sub-Plane in the realization of the vertical UCs.

Table 1 Summary of cognition-based management per SliceNet Use Case

Use Case	Short description
5G Smart Grid Self-Healing	The Smart Grid Self-Healing UC focuses on validating an advanced self-healing solution for electric power grids. The goal is to minimize the service downtime. Two major issues are involved: 1) detecting and isolating the electric failure, 2) efficiently reconfiguring the power grid.
5G e-Health Connected Ambulance	The e-Health UC focuses on a connected ambulance that acts as a connection hub for the emergency medical equipment and wearables, enabling storing and real time streaming of video data to the awaiting emergency department team at the destination hospital.
5G Smart City	The Smart City UC focuses on managing an intelligent public lighting system. This system will enable the control of every single pole and the monitoring of the current consumption. As such, it will give authorities the ability to turn the lights on and off remotely but also to dim them according to a schedule, reducing the overall power consumption of the public lighting system.

In the following sub-sections, a brief summary of the UCs is reported. In particular, the role of the Cognition Sub-Plane is highlighted.

2.2 5G Smart Grid Self-Healing Use Case

2.2.1 Description

The key objective of the Smart Grid (SG) UC is to support a self-healing solution for electric power grids. From the power grid perspective, two major steps are involved - the first one is on detecting and isolating the electric failure, whereas the second one is to efficiently reconfigure the power grid and therefore minimize the service downtime.

Intelligent Electronic Devices (IEDs) are used to sense the electric power grid and guarantee that it is configured accordingly and rapidly. To achieve this, the IEDs require a reliable Radio Access Network (RAN) connectivity service to allow ultra-low latency (~15 ms) communications between them. A specific network communication protocol - IEC 61850 GOOSE - is used to establish the IEDs communications in order to protect and reconfigure the power grid. Besides the inter-IEDs communications for the power grid protection and reconfiguration, this scenario also requires a communication service between the IEDs management system and the IEDs for management purposes, such as configuration, remote diagnosis, etc.

From a business point of view, the vertical will subscribe the E2E services (Ultra Low Latency Service and Management Service) from a Digital Service Provider (DSP), indicating the service's endpoints (IEDs and SG Management Server), as well as the associated requirements. Associated with each service subscription is a Service Level Agreement (SLA) which indicates the delivery obligations of the DSP, as well as the counter-measures if there is a violation. It is fully abstracted from the vertical how exactly the DSP delivers the contracted services, as well as the involved Network Service Providers (NSPs). Figure 1 provides an overall perspective of the SG UC, including the vertical devices (in blue), the DSP actor (in red) and the subscribed E2E services - Ultra-Low Latency (ULL) Communication Service (dashed line) and the Management Communication Service (continuous line).

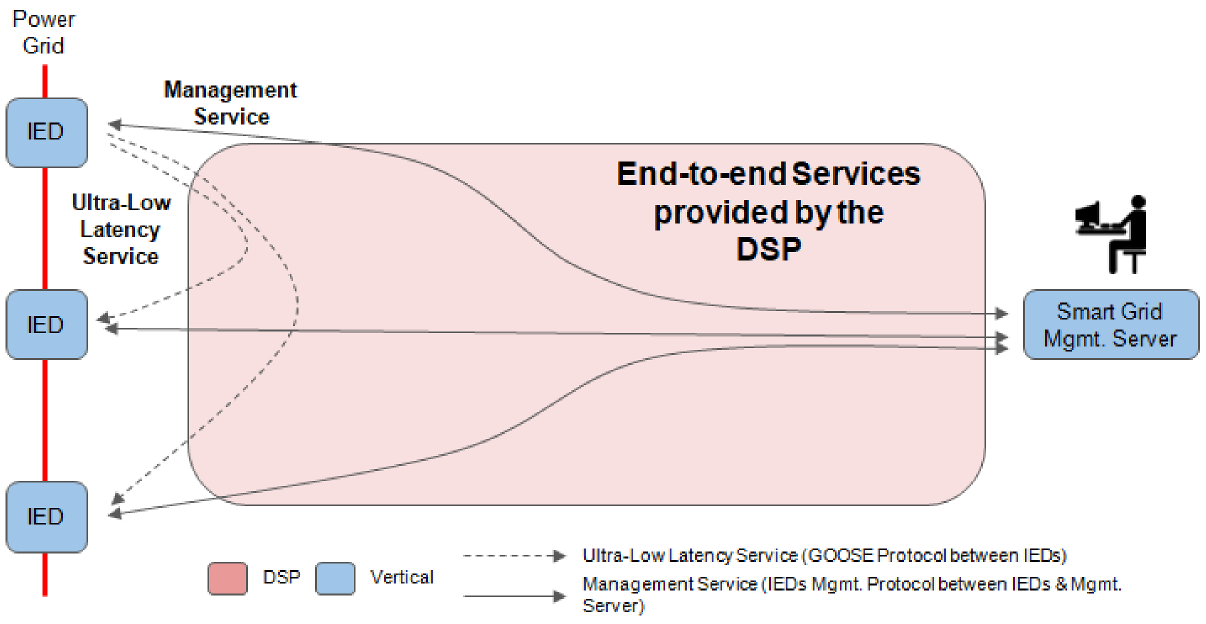

Power Grid
IED
IED
IED
Management Service
Ultra-Low Latency Service
End-to-end Services provided by the DSP
Smart Grid Mgmt. Server
DSP
Vertical
Ultra-Low Latency Service (GOOSE Protocol between IEDs)
Management Service (IEDs Mgmt. Protocol between IEDs & Mgmt. Server)

Figure 1 Smart Grid Use-Case High Level Overview

For the ULL Communication Service subscribed by the Vertical, an E2E Ultra Low Latency (E2E ULL) Slice is used, as depicted in Figure 2. A complete 5G mobile network is required to deliver this E2E slice - including RAN and Mobile Core components. The slice endpoints are the Subscriber Identification Module (SIM) cards connected to the IEDs 5G modem.

The E2E slice is a logical concept, managed by the DSPs, which is materialized in NSes instantiated at each NSP (which is the business entity managing the NS physical/virtual resources). In this particular UC, due to the IEDs geographic distribution, the E2E ULL Slice is instantiated using RANs from two NSPs (but a common Mobile Core). Shortly, in terms of decomposition, the E2E ULL Slice, illustrated in Figure 2 (grey colour), is translated in the following NSes:

1. RAN + Mobile Core NS **@NSP1** (green colour);
2. RAN NS **@NSP2** (yellow colour).

It is important to mention that the E2E ULL slice (subscribed by the vertical) translation to the NSP NSes is transparent to the vertical. In other words, depending on the vertical requirements and on the available NSP NS offers, the DSP decides, during the E2E slice instantiation phase, which NSP NSes will be required. Therefore, from the business perspective, the vertical only "sees" the SLA with the DSP. On the other hand, the DSP must manage contractual relationships with the NSP (or NSPs) providing the NSes. Further details about the SG UC are provided in D5.6 [2] and D7.2 [3].

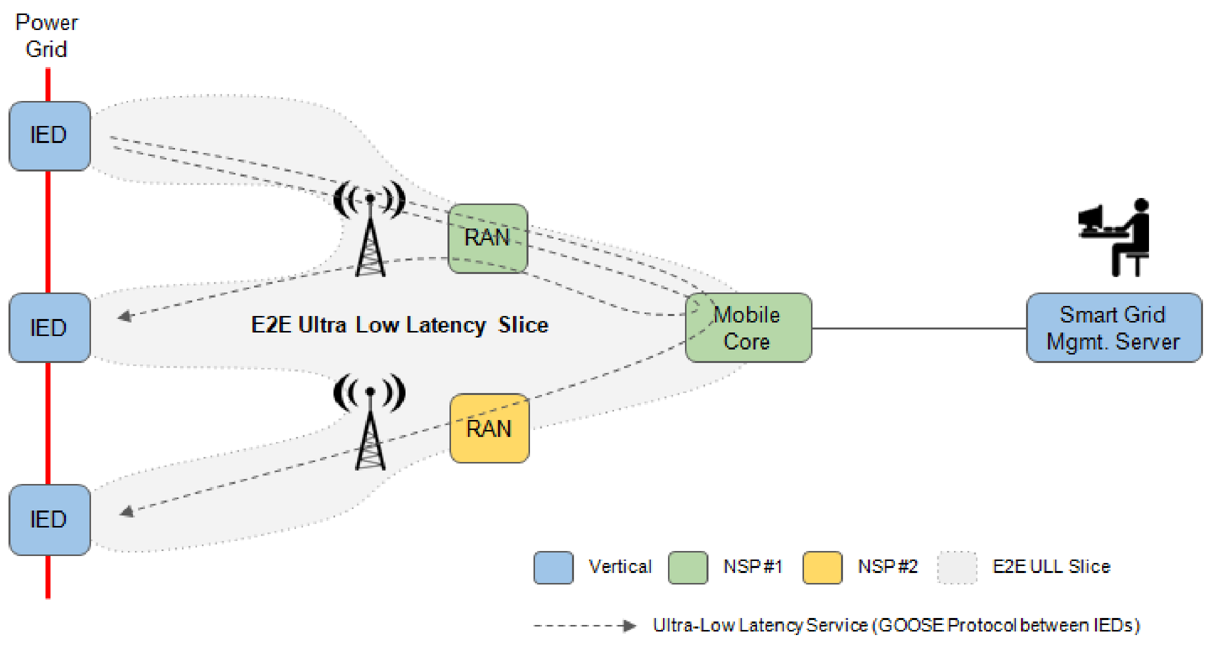


Figure 2 Smart Grid Use-Case - E2E Ultra Low Latency Slice

The DSP must be able to guarantee that the SLA obligations to the vertical are met. In this case, for the E2E ULL slice, it should guarantee that the IED - IED communications over the 5G mobile network delivers **latencies below 10 ms** and that **no packets are lost**. On the other hand, the NSPs must be able to guarantee to the DSP that the RAN + Mobile Core NS (**@NSP1 –** green colour) and the RAN NS (**@NSP2** – yellow colour) requirements are also met.

The SliceNet Cognition Sub-Plane is critical for the NSPs and the DSP in order to guarantee that the contracted SLAs are met during the whole slice's lifetime. At the **NSP level**, the Cognition Sub-Plane is responsible for predicting RAN faults and therefore avoiding an unreliable NS to be delivered towards the DSP. Proactively detecting network faults will enable their mitigation within the NSP borders, without affecting the SLA towards the DSP, as well as without impacting the vertical QoE. At the **DSP level**, the Cognition Sub-Plane is responsible for guaranteeing that the E2E NS respects the contracted SLA. In the SG UC, the Cognition Sub-Plane at the DSP is alerted when the NSP is not able to meet the contracted SLAs and, as a mitigation action, decides which are the best candidate NSP(s) to replace the imminent faulty NSP.

The following sub-sections will detail the role of the Cognition Sub-Plane at the NSPs and DSP, respectively.

2.2.2 NSP Instantiation

As described in the previous sub-section, the Cognition Sub-Plane is responsible for predicting RAN faults at the NSP level. Figure 3 illustrates the instantiation of the NSP for the SG UC, depicting the cognition workflow, which is responsible for real-time prediction of RAN-related alarms.

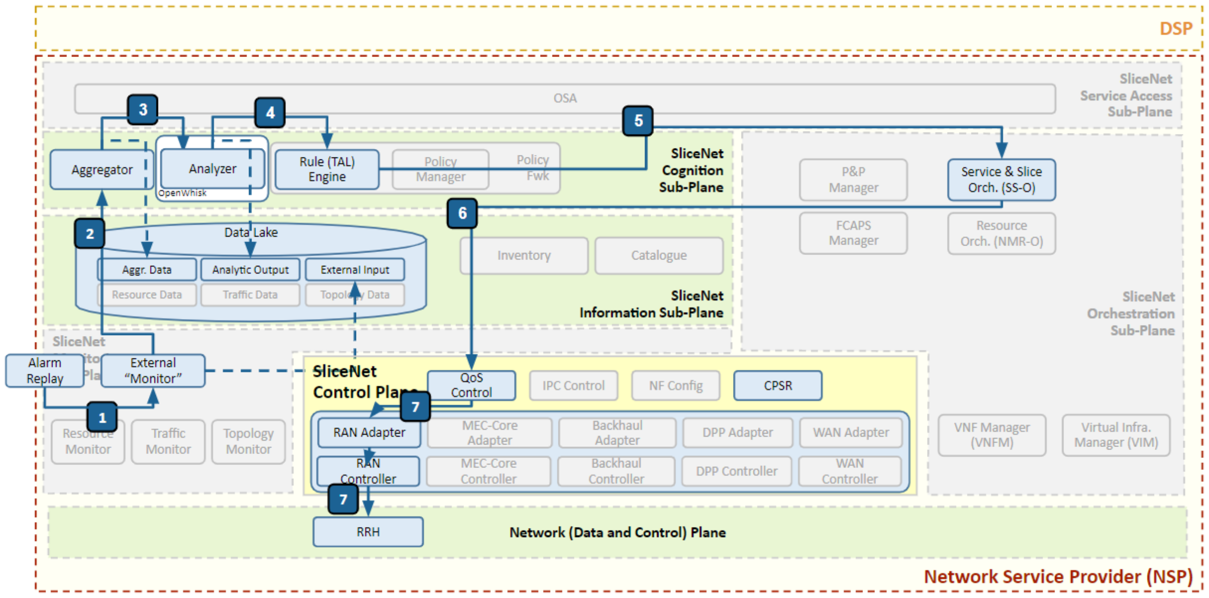


Figure 3 SliceNet Smart Grid UC instantiated at the NSP

Table 2 describes the SG UC steps implemented in the MAPE/cognition workflow.

Table 2 SG UC MAPE/cognition workflow (at NSP) steps description

Step	Description
1	In this UC, the alarms data is provided from Altice Network Operator (NO) from Portugal (named MEO). Therefore, the training and online predictions are made over the MEO alarms data, whereas the actuations are made directly on the SG UC testbed using the Open Air Interface (OAI) RAN. The MEO data is replayed and ingested to the SliceNet system through the **External Monitor** component. The data is collected and thereafter delivered to the **Aggregator** in the next step. In parallel, the **External Monitor** also persists the collected data on the **Data Lake**.
2	The alarms data from MEO are delivered to the **Aggregator**, which is responsible for the preparation and transformation procedures in real-time. When completed, the Aggregator streams the data towards the Alarms Prediction ML model running in the **Analyser**. Besides streaming the data, the **Aggregator** also persists the data in the **Data Lake**.
3	The **Analyser**, in which the Alarms Prediction ML model is running, consumes the transformed data in real-time and runs the model to produce the alarms insights/predictions. The produced alarms insights/predictions are stored in the **Data Lake** and streamed for real-time consumption and reaction.
4	The **Rule (TAL) Engine** consumes the alarms prediction **E**vent, checks the configured NSP policies **C**onditions, in this case the slice available bandwidth, and as a mitigation **A**ction decides to increase the NS bandwidth to avoid the network fault. The **Rule (TAL) Engine** implements the **ECA** (**E**vent – **C**ondition – **A**ction) policy approach.
5	The action plan decided by the **Rule (TAL) Engine** is delivered to the **Service & Slice Orchestrator**.
6	Following the NS bandwidth increase request issued by the **Rule (TAL) Engine**, the **Service & Slice Orchestrator** interacts with the **Quality of Service (QoS) Control** component at the SliceNet Control Plane to enforce the NS bandwidth modification.
7	The **QoS Control** identifies the appropriate network segment (RAN) adapter to address the bandwidth increase request and delivers it to the **RAN Adapter**. The **RAN Adapter** delivers the request to the **OAI RAN Controller**, which translates the generic bandwidth increase information request to OAI RAN specific parameters. Finally, the **RAN Controller** modifies the

	NS bandwidth by interacting with the OAI Remote Radio Head (RRH).

2.2.3 DSP Instantiation

On the DSP side, the Cognition Sub-Plane is responsible for selecting the most appropriated NSPs to be engaged in the E2E NS, guaranteeing that the E2E service delivered to the vertical fulfils the contracted SLA. Figure 4 illustrates the instantiation of the DSP for the SG UC, depicting the cognition workflow.

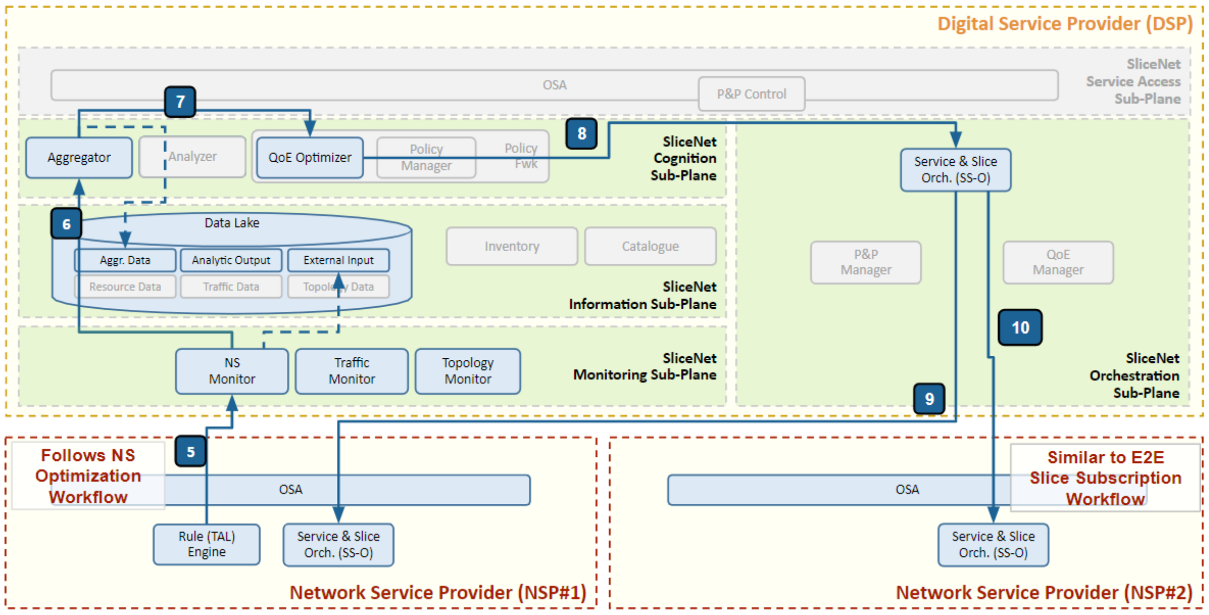

Digital Service Provider (DSP)
OSA
P&P Control
SliceNet Service Access Sub-Plane
7
8
Aggregator
Analyzer
QoE Optimizer
Policy Manager
Policy Fwk
SliceNet Cognition Sub-Plane
Service & Slice Orch. (SS-O)
6
Data Lake
Aggr. Data
Analytic Output
External Input
Resource Data
Traffic Data
Topology Data
Inventory
Catalogue
SliceNet Information Sub-Plane
P&P Manager
QoE Manager
10
NS Monitor
Traffic Monitor
Topology Monitor
SliceNet Monitoring Sub-Plane
9
SliceNet Orchestration Sub-Plane
5
Follows NS Optimization Workflow
OSA
Rule (TAL) Engine
Service & Slice Orch. (SS-O)
Network Service Provider (NSP#1)
Similar to E2E Slice Subscription Workflow
OSA
Service & Slice Orch. (SS-O)
Network Service Provider (NSP#2)

Figure 4 SliceNet Smart-Grid UC instantiated at the DSP

Table 3 describes the SG UC steps implemented in the MAPE/cognition workflow.

Table 3 SG UC MAPE/cognition workflow (at DSP) steps description

Step	Description
5	The **NS Monitor** collect NSes information related with the imminent faulty NS at NSP1.
6	The collected information is delivered to the **Aggregator** which will perform the required preparation and transformation procedures on the data, create E2E NS perspective metrics and stream the result towards the **QoE Optimizer**.
7	The **QoE Optimizer** consumes the event produced by the Aggregator and decides, based on policies delivered by the Policy Manager, that NSP1 RAN NS should be replaced by NSP RAN NS.
8	The final decision is delivered to the **Slice & Service Orchestrator** of the DSP, which will enforce the required actions to replace NSP1 RAN NS by NSP2 RAN NS.
9	The **Slice & Service Orchestrator** requests NSP1 to decommission the RAN NS.
10	The **Slice & Service Orchestrator** requests NSP2 to instantiate the RAN NS. This procedure is described in D7.2 [3].

2.3 5G e-Health connected Ambulance Use Case

Previously, in deliverable D5.6 [2], an anomaly detection model was proposed and tested on real data with high performance. In this deliverable, we will detail its integration within the SliceNet framework. This section will present the scenario and the integration details.

2.3.1 Description

The SliceNet eHealth UC aims to provide a NS for the vertical (hospital) offering the best services and an excellent QoE to a paramedic team. A smart connected ambulance is roaming through the NSPs while sending data streams that should be delivered in real time with no quality degradation. A feedback mechanism is implemented, allowing for verticals to express their perceived quality of the service every second. In this regard, the focus of the developed ML model is to serve as an intelligent QoE sensor by analyzing these data samples. The Cognitive Sub-Plane's anomaly detection model for the eHealth UC's goal is to predict if a degradation in the network signal strength in the RAN segment may be perceived in the future 5 minutes by observing the last 5 minutes of QoS metrics that are perceived by the vertical. A streaming of the vertical feedback will be sent every 5 seconds to the One Stop Application Programming Interface (API) (OSA), which, in collaboration with the P&P controller, will deliver the data towards the Cognition Sub-Plane. The used KPIs are well-known QoS metrics in RAN domain, which are detailed in Table 4. The Cognitive Sub-Plane identifies that the signal quality will be degraded and actuates a handover in order to maintain optimal QoE levels.

Table 4 e-Health Use Case related QoS metrics

KPI	Description
Timestamp	The timestamp of the measurements
IMSI	The International Mobile Subscriber Identity (IMSI) of the User Equipment (UE) mobile device sending data
Perceived Reference Signal Receive Power	The average power received from a single reference signal
Perceived Reference Signal Receive Quality	It indicates quality of the received signal
Perceived Signal to Noise Ratio	It measures the signal strength relative to background noise.
Perceived Channel Quality Indicator	It is an indicator carrying the information on how good/bad the communication channel quality is
Perceived Received Signal Strength Indication	Signal strength measured from all base stations
Perceived Downlink bitrate	The bit rate in down link
Perceived Uplink bitrate	The bit rate in up link

The trained anomaly detection model allows predicting the signal quality in the future 5 minutes. Once the model predicts low signal strength for an urgent video stream, the Actuation Framework within SliceNet's Cognition Sub-Plane will trigger the event in order to do a remedial action and solve the problem before it occurs. One possible actuation is to do a handover of the Evolved Node B (eNB) to which the ambulance is connected to another eNB in the same NSP. This model also offers the opportunity for the vertical to supervise the performance of its E2E NS and to express its feedback to Slicenet; for its part, the SliceNet framework will react with some remedial actions.

2.3.2 DSP Instantiation

The cognition capabilities that allow for the realization of the eHealth vertical UC and, more concretely, the Anomaly Detection cognition technical UC (TUC), reside exclusively at the DSP level. In this regard, the DSP is responsible for maintaining the quality of the ambulance connection to the provided slice in order to guarantee a healthy E2E service across the provisioned infrastructure. After collecting RAN-related information from the UE endpoint, the DSP determines if an anomaly is going to happen at the connection channel that may affect ongoing eHealth service. This being the case, the DSP triggers the necessary remedial actions to overcome the predicted anomaly.

In this regard, two modes of operation have been defined for the cognitive-based loop. One mode, labelled as a **streaming** mode, has been defined in which data coming from the UE is directly fed onto the analytical function responsible for detecting anomalies for real-time analysis of the collected data. The other mode, labelled as **offline** mode, differs in the fact that the data from the UE is stored at the DSP shared Data Lake, which then aggregates the several measurements into a Comma Separated Values (CSV) file for its latter consumption by the analytical model. Figure 5 and Figure 6 illustrate the instantiation of the eHealth UC, depicting the cognition workflow for both operation modes, respectively.

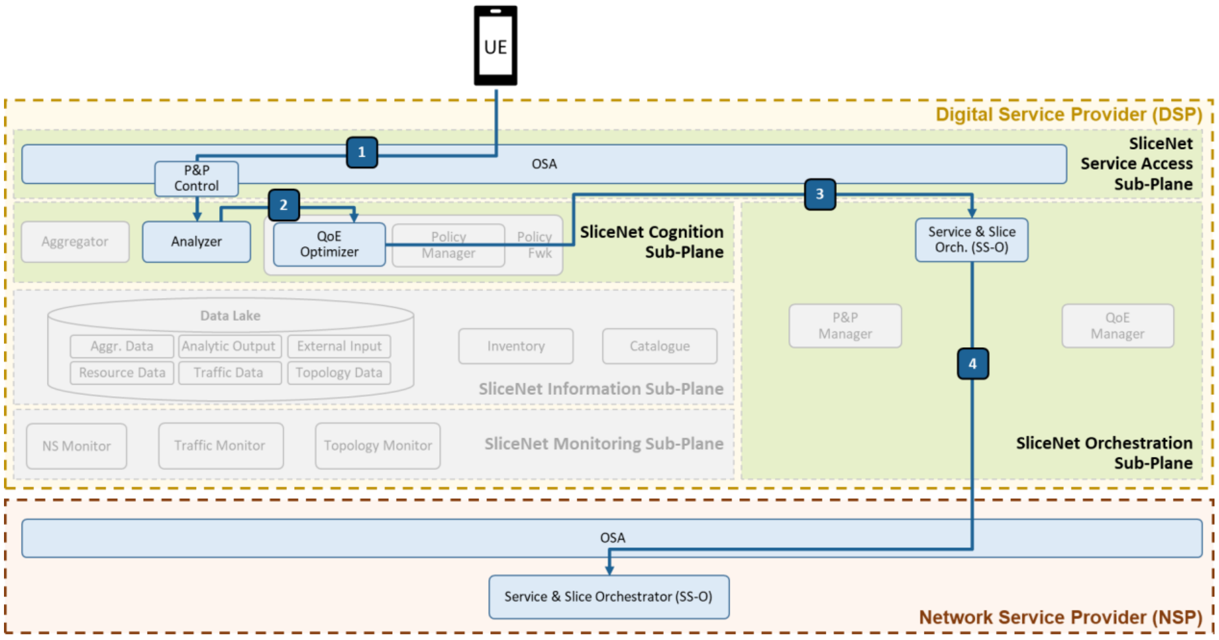

UE
Digital Service Provider (DSP)
SliceNet Service Access Sub-Plane
OSA
P&P Control
1
2
3
4
Aggregator
Analyzer
QoE Optimizer
Policy Manager
Policy Fwk
SliceNet Cognition Sub-Plane
Service & Slice Orch. (SS-O)
Data Lake
Aggr. Data
Analytic Output
External Input
Resource Data
Traffic Data
Topology Data
Inventory
Catalogue
SliceNet Information Sub-Plane
P&P Manager
QoE Manager
NS Monitor
Traffic Monitor
Topology Monitor
SliceNet Monitoring Sub-Plane
SliceNet Orchestration Sub-Plane
OSA
Service & Slice Orchestrator (SS-O)
Network Service Provider (NSP)

Figure 5 SliceNet eHealth UC instantiated at the DSP – streaming mode

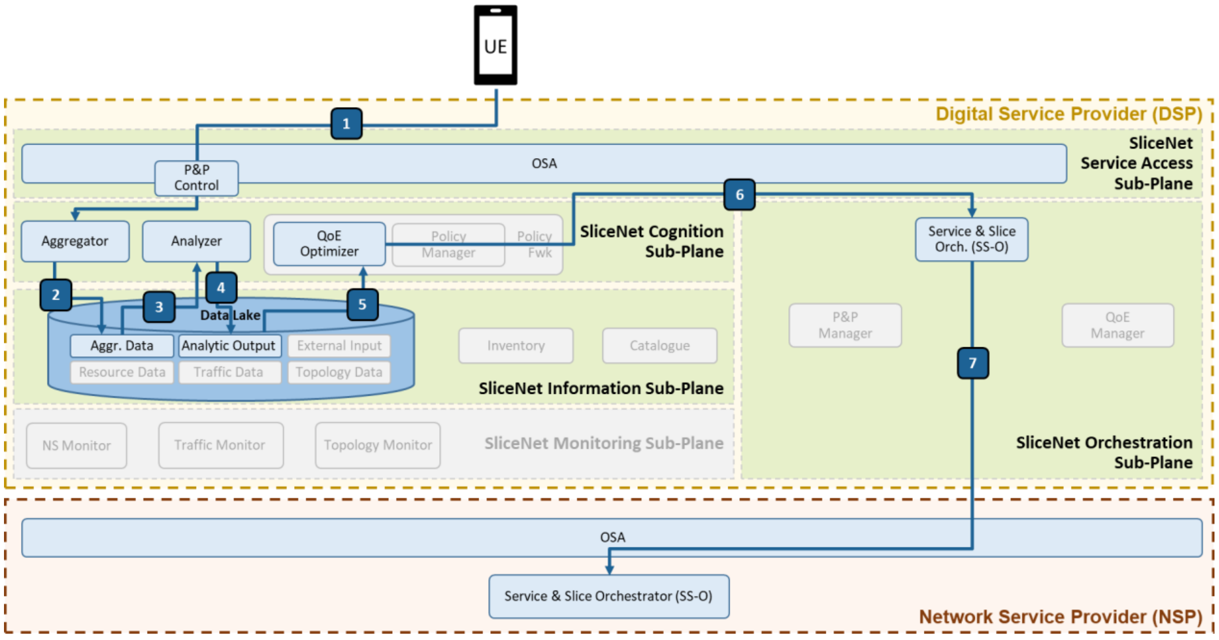

UE
Digital Service Provider (DSP)
SliceNet Service Access Sub-Plane
OSA
P&P Control
1
2
3
4
5
6
7
Aggregator
Analyzer
QoE Optimizer
Policy Manager
Policy Fwk
SliceNet Cognition Sub-Plane
Service & Slice Orch. (SS-O)
Data Lake
Aggr. Data
Analytic Output
External Input
Resource Data
Traffic Data
Topology Data
Inventory
Catalogue
SliceNet Information Sub-Plane
P&P Manager
QoE Manager
NS Monitor
Traffic Monitor
Topology Monitor
SliceNet Monitoring Sub-Plane
SliceNet Orchestration Sub-Plane
OSA
Service & Slice Orchestrator (SS-O)
Network Service Provider (NSP)

Figure 6 SliceNet eHealth UC instantiated at the DSP – offline mode

The following tables describe the eHealth UC steps implemented in the MAPE/cognition workflow for both operation modes, namely, streaming mode (Table 5) and offline mode (

Table *6*).

Table 5 eHealth UC MAPE/cognition workflow (at DSP) steps description – streaming mode

Step	Description
1	Thanks to the capabilities exposed through the **OSA** and the **P&P Control**, the information related to quality metrics from the **UE** is directly streamed towards the **Analyser**.
2	The **Analyser** (i.e. the Anomaly Detection ML model) analyses the data and produces an anomaly prediction event (if any) which is sent directly to the **QoE Optimizer**.
3	The **QoE Optimizer** consumes the event and decides, based on policies delivered by the Policy Manager, that in order to avoid predicted anomalies onto the ongoing eHealth service, a handover is required. A handover means that the UE is re-associated with a healthier eNB or RAN slice within the NSP infrastructure and disassociated from the failing slice's resources. The decision is delivered to the **Slice & Service Orchestrator** of the DSP.
4	The **Slice & Service Orchestrator** at DSP level engages with the **Slice & Service Orchestrator** at NSP level in order to enforce the handover/UE re-association. The exact action will be carried out thanks to NSP level capabilities (e.g. Control Plane functions).

Table 6 eHealth UC MAPE/cognition workflow (at DSP) steps description – offline mode

Step	Description
1	Thanks to the capabilities exposed through the **OSA** and the **P&P Control**, the information related to quality metrics from the **UE** is collected by the DSP **Aggregator**.
2	The **Aggregator** performs the required preparation and transformation procedures on the data to create a CSV file containing several UE measurements, and insert it onto the DSP **Data Lake**.
3	The aggregated CSV file is collected by the **Analyser** from the shared **Data Lake**.
4	The **Analyser** (i.e. the Anomaly Detection ML model) analyses the data and produces an anomaly prediction event (if any) which is inserted back as elaborated data to the DSP **Data Lake**.
5	The prediction events are polled from the **Data Lake** by the **QoE Optimizer** to determine if remedial actions are required.
6	The **QoE Optimizer** decides, based on policies delivered by the Policy Manager, that in order to avoid the predicted anomalies onto the ongoing eHealth service, a handover is required. A handover means that the UE is re-associated with a healthier eNB or RAN slice within the NSP infrastructure and disassociated from the failing slice's resources. The decision is delivered to the **Slice & Service Orchestrator** of the DSP.
7	The **Slice & Service Orchestrator** at DSP level engages with the **Slice & Service Orchestrator** at NSP level in order to enforce the handover/UE re-association. The exact action will be carried out thanks to NSP level capabilities (e.g. Control Plane functions).

The two developed modes serve different interests and purposes. On one hand, the **streaming** mode enables the real-time processing of the monitored UE data, its subsequent analysis and near-real-time actuation to overcome the predicted anomalies. This approach is essential in scenarios in which critical services, as it is the case of the eHealth vertical UC, are being executed, since they require a fast and up-to-date analysis and actuation to guarantee optimal quality levels. However, this real-time process requires high computational capabilities to be realized, potentially being overtaxing in situations with large amounts of data.

In light of this, the other developed mode, **offline**, relaxes the time constraints by making use of the shared Data Lake. In such a case, the data is not processed in real-time but rather processed in batches, which eases the burden of the analytical functions. This approach is suitable for services instances whose priorities are lower and thus less stringent times may be applied for the analysis and actuation processes. In addition, the **offline** mode may execute background analysis that can help and complement the capabilities of the **steaming** mode. All in all, the two developed modes are not exclusive, on the contrary, they complement each other, serving specific goals within the overall vertical UC.

2.4 5G Smart City Use Case

2.4.1 Description

Previously, in deliverable D5.6 [2], a ML model highlighting the Noisy Neighbour (NN) issue detection and mitigation process was proposed and tested on the Orange testbed infrastructure. The noisy neighbour problem describes a cloud computing multi-tenant infrastructure that consumes all bandwidth, disk, Central Processing Unit (CPU) and other resources, negatively impacting other user's performance. Since the effect of noise causes issues to others Virtual Machines (VMs) and applications that are using the same infrastructure cloud network (analogous to multiple slices sharing the same NSP infrastructure), it has been identified as problem that would affect the Smart City UC performance.

The Smart City UC is based on a NS provided by SliceNet infrastructure for the Smart Lighting application. In the real scenario, the NS ensures connectivity for tens of thousands of connected devices, across the virtualized network environments. As the UC requirements are not highly related to metrics (latency, delay, packet loss, or bandwidth per device) and the UC information – Internet of Things (IoT) packets – is not critical in terms of QoS metrics, the UC's performance may be more impacted by its sharing of infrastructure network resources, as one of the use case's VMs may work in a noisy infrastructure environment. Table 7 provides a summary of the main QoS metrics considered for the Smart City UC.

Table 7 Smart City service QoS metrics

QoS metric	QoS Range
Service availability	99,99 %
Bandwidth	20-100 kbps
Latency	50-300 ms
Packet loss	<=0.1%

In addition, the list below provides the format of the IoT packets for the traffic flows experienced within the UC:

- Size ≈ 1408B
- IoT Periodicity: one packet/minute
- Transport layer: User Datagram Protocol (UDP)
- Application protocol: Message Queuing Telemetry Transport (MQTT)
- Command on/off: 70B
- Capacity/device: 50kbps

The UC demonstrates the ability of applying cognitive methods for QoS metrics analysis and to predict the status of Virtual Network Function (VNF) instances running in a Network Function Virtualization Infrastructure (NFVI) environment. Noise in the virtualized environment (overload of the compute node that is hosting the UC's VNF for IoT) is monitored, by the infrastructure collected data, providing the capability to detect and predict that a specific VNF is affected by the infrastructure noise generated by other resource consumers.

In order to demonstrate the capabilities of the Cognition Sub-Plane for the Smart City UC, a real scenario demonstration has been executed within Orange testbed (Figure 7). In particular, two NSes have been deployed, with the first NS being the Smart City UC slice whereas the second NS plays the role of another consumer of the virtualized infrastructure, generating network traffic and using infrastructure resources, thus simulating the noise over the Smart City UC slice. The IoT VM and the Video VNF are deployed on the same compute node. Due to the compute node's excess resource consumption (other VNFs are consuming the server CPU) the IoT platform experiences processing delays issues. The Cognitive Sub-Plane identifies the problem and actuates the remedial action to reduce the overload condition thus ensuring the desired QoE.

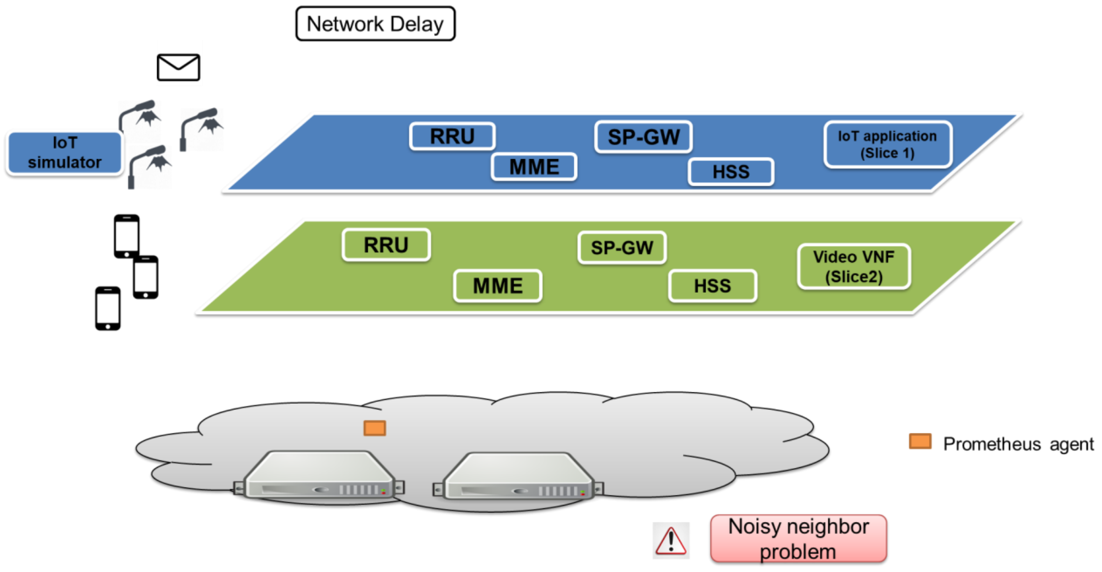


Figure 7 Smart City concurrent NSes competing for resources creating the noisy neighbour problem

In such environment, the SliceNet Cognition Sub-Plane should guarantee that the contracted SLAs are met during the slice lifetime, keeping the slice QoE at optimum levels. The following sub-section details how the Cognition Sub-Plane is exercised for the Smart City UC.

2.4.2 DSP/NSP Instantiation

The Smart City UC is unique in that it demonstrates the case in which both administrative roles (DSP and NSP) collapse onto the same entity, which then assumes the management and control capabilities offered by the multiple components within the SliceNet architecture. As a result, a hybrid instantiation of the Cognition Sub-Plane is materialized, in which, instead of having a separate MAPE-K/cognition loop per administrative role, a unique loop spans all the functionalities starting from the 5G network infrastructure towards the capabilities of the Cognition Sub-Plane. For example, a single Data Lake is instantiated, which holds all the data of the full layered system. Additionally, actuations related to quality of deployed slices do not require the engagement of both Policy Decision Points (PDPs), i.e. the Rule (TAL) Engine and the QoE Optimizer, but is enough that all QoE-related (re-)configurations are handled by the QoE Optimizer. Given these details, Figure 8 illustrates the instantiation of the Cognition Sub-Plane and the related MAPE-K loop for the Smart City UC assuming a single administrative entity.

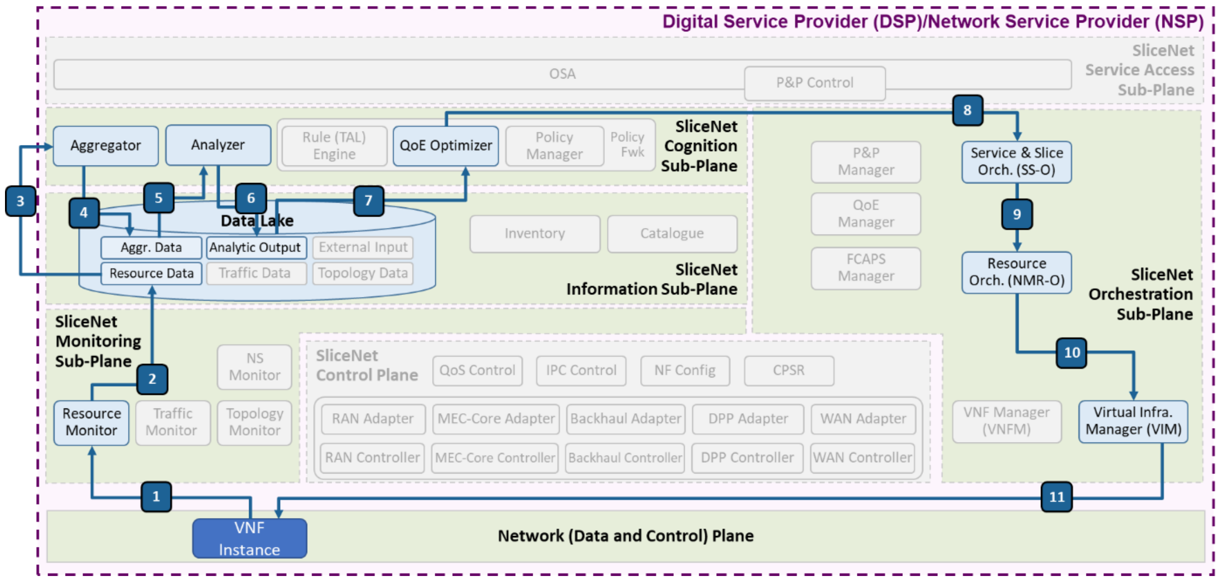


Figure 8 SliceNet Smart City UC instantiation

Table 8 describes the Smart City UC steps implemented in the MAPE/cognition workflow.

Table 8 Smart City UC MAPE/cognition workflow steps description

Step	Description
1	Performance metrics related to the **VNF instances** belonging to the Smart City E2E slice are collected thanks to the **Resource Monitor** at the Monitoring Sub-Plane.
2	The **Resource Monitor** stores the collected metrics into the **Data Lake**.
3	This **Resource Data** is elaborated by the **Aggregator,** which prepares and transforms the data for it to be ML-ready.
4	The **Aggregator** inserts the aggregated data into the shared **Data Lake**.
5	The aggregated data related to the performance of the deployed VNFs is collected by the **Analyser** from the **Data Lake**.
6	The **Analyser** (i.e. the NN ML model) analyses the data and produces an event which indicates if any of the monitored/supervised VNFs supporting the IoT applications of the Smart City service is under any undesired state (e.g. overloaded, noisy). These events are the inserted back into the **Data Lake**.
7	The **QoE Optimizer**, acting as the sole PDP, polls the **Data Lake** and collects the stored events related to the previous analysis.
8	The **QoE Optimizer** decides, based on policies delivered by the Policy Manager, the most suitable remedial action to overcome the undesired states of affected VNF instance. It may decide on scaling overloaded VNFs or migrations related to quieting the noisy VNFs (lowering their priorities, reducing their allotted resources, etc.). The decision is delivered to the **Slice & Service Orchestrator**.
9	The **Slice & Service Orchestrator** delivers the decision to the **Resource Orchestrator**.
10	The **Resource Orchestrator**, given the forwarded request and its parameters, determines which is the extra resources that are required (upscaling case) or the new location (migration case) for the VNF instance for which the actuation has been triggered. To enforce the desired action, it engages with the **Virtual Infrastructure Manager**.
11	The **Virtual Infrastructure Manager**, being responsible for the management of the virtual computation resources (e.g. VMs), executes the determined action.

3 Progress to Analytic Workflows

This section discusses the progress to the analytic workflows described in D5.6 [2]. The implemented analytic workflows demonstrate the logical SliceNet workflows responsible for predicting NS QoE KPIs, including obtaining the required data, learning, processing, and providing the necessary information for actuation, to allow NS QoE management. These workflows enrich the raw data from multiple sources (including the vertical), adding insights and predictions to support vertically-informed actuations.

The implementations exercise several E2E logical workflows of SliceNet that interact with the Cognition Sub-Plane (see Table 9). We demonstrate monitoring and cognition at both network sub-slice (NSS) and E2E NS/service levels, providing the building blocks for the implementation of Vertically-Informed QoE Sensors. The focus at this stage is on the learning phase of the cognitive pipeline, validating the applicability of the analytic methods employed using external and simulated data. New to the current iteration, the several presented analytic workflows provide expanded results sections, providing experimental evaluations of the several cognition TUCs that have been developed, and linking them to the corresponding actuation workflows.

Table 9 Summary of analytic prototype workflows

Name	**Short description**
Reliable RAN slicing using NSP alarm data	Demonstrate processing of external data sources to support NS reliability; optimizing resource selection during NS creation and predicting imminent failures.
Noisy Neighbour detection	Demonstrate the ability to provide a QoE sensor that monitors slice-level metrics and applies cognitive methods to predict service level degradation and pinpoint its origin (application vs. provider).
QoE classification from QoS metrics	Demonstrate the usage of vertical feedback at the training stage to develop a model for predicting E2E QoE from measured QoS KPIs.
RAN optimisation	Demonstrate the application of cognitive methods for managing the RAN effectively and optimizing its capacity to maintain high QoS for multiple slices and meet their desired service-based performance objectives.

3.1 Reliable RAN slicing using NSP alarm data

One of the requirements of having a real-time prediction model is the creation of a process that evaluates how good these predictions are when deployed. Every model has its synthetic metrics that evaluate the performance after the training, but since the model is generated from historical data, there must be a continuous re-evaluation of the model to ensure against the degradation of its prediction quality over time. Model supervision introduces a whole set of new concepts on how to evaluate model performance, which are described in the next sub-sections.

Another dimension of this study is the translation of model prediction events and how they are related to alarm instances. Remember that the original MEO dataset is made of events of instances. When deployed and in runtime production mode, the model reads and creates events that must be translated back into instances, as they carry the semantic meaning required to apply corrective actions in the network.

The deployment of the models will result in the output of predictive events, when evaluating, these events cannot be mapped directly on decisions made in the Policy Framework (PF), and as such some form of transformation is required as the main element to be interpreted in PF actuations. To this effect, the events are mapped into instances with a lifetime limit equal to the model's prediction interval (6h at the time of this writing), if new events arrive this limit is updated and extended. Extending the predictions to alarm instances leverages the semantic in the data. Interpreting events directly would distort the results and therefore the required actions. This section explains how the output of the predictive models generates predictive alarm instances and how it relates to the real alarm instances. The terms and used definitions are described in Table 10.

Table 10 Terms and Definitions in the prediction model output

Term	Definition
Predictive event	Output of the model, can be a positive prediction or a negative one
Predictive alarm instance/predictive instance	The instance generated by using the predictive events
Real alarm instance/alarm instance	the instances created from events received from the equipment or other systems

3.1.1 Context and properties of the data

Each record in the input data represents an event. The event may update the information of the state of the system by starting, modifying or ending instances. We can say that each state is defined by an event, while the event alone does not define the state. The state of a given moment contains information regarding the instances that are open. In particular, an instance has the following properties:

- *specificProblem* → what is the problem associated with the instance.
- *start time* → when the instance started.
- *end time* → when the instance was closed (might be undefined if the instance is open).
- *last event* → when was the arrival of the last event associated with this instance.

In addition, a state has the following properties:

- *Instances* → a set of open instances.
- *localcode* → scope of the location of the events that defined this state.
- *timestamp* → the time when this state was defined

The prediction model takes a state as input and outputs a prediction. This means that each prediction is defined from a state. The model's output has the following properties:

- *specificProblem*
- *localCode*
- *timestamp*
- *likelihood of occurrence*

The evaluation of the predictions is made by matching the models output with the target instances set. The target instances are instances that are closed (the end time property is defined) and where the *specificProblem* belongs to a given set of target specific problems. When reading the model output, a given *threshold* associated with the likelihood of occurrence classifies the output as a **positive** event prediction - predicts that a target instance will happen - or **negative** event prediction - predicts that a target instance won't happen. By joining the model output with the target instances with a predefined **target time** we can define the event predictions as being **true** or **false**.

A positive event prediction is classified as **true positive** if the target instance is occurring as predicted. A **false positive** classification means that the target instance is missing within the set target time (the prediction was wrong) or that the instance is occurring when the prediction is made (no actual prediction was made).

For the **negative** event predictions, the opposite conditions apply: if there is a target instance within the set target time, the prediction from the model output is defined as a **false negative**, otherwise as a **true negative**. Note that since each prediction came from a state and each state came from an event, we can say that this is a unit associated with the prediction of a target instance.

For the evaluation, we will define **Δt** as the time to the start of the next target instance of this prediction unit. This Δt is a property of the relation between the prediction and the next target instance. Since several predictions can precede a single target instance, each target instance can have multiple predictions associated. The evaluation of the models will be focused on the instances. Figure 9 depicts a schematic of the process for successfully evaluate a prediction event.

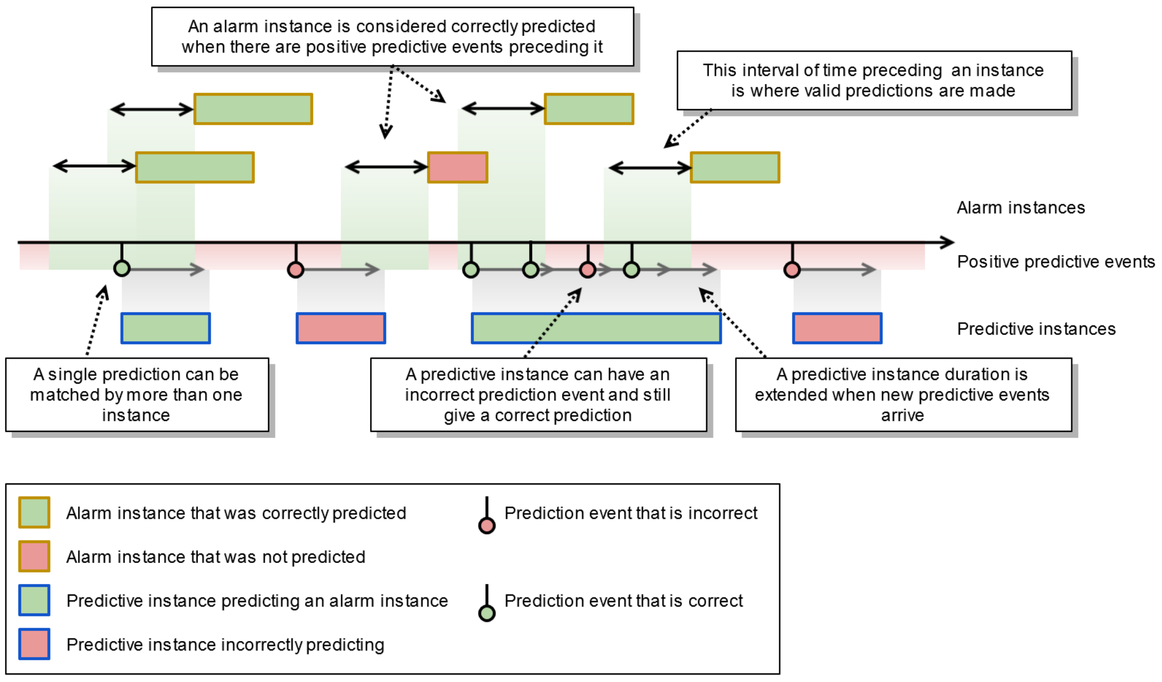

An alarm instance is considered correctly predicted when there are positive predictive events preceding it
This interval of time preceding an instance is where valid predictions are made
Alarm instances
Positive predictive events
Predictive instances
A single prediction can be matched by more than one instance
A predictive instance can have an incorrect prediction event and still give a correct prediction
A predictive instance duration is extended when new predictive events arrive
Alarm instance that was correctly predicted
Alarm instance that was not predicted
Predictive instance predicting an alarm instance
Predictive instance incorrectly predicting
Prediction event that is incorrect
Prediction event that is correct

Figure 9 Success evaluation of predictive events

3.1.2 Evaluations

Let the evaluated target instances be defined by the following sets:

- S = {all target instances}
- FN = {all target instances associated with false negative (FN) predictions}
- TP = {all target instances associated with true positive (TP) predictions}
- M = S ∩ ¬(FN ∪ TP) = {all target instances with no associated predictions}

Prediction of instances is considered for a given period before their start. A prediction can be **Positive** 🡑 (instance occurs) or **Negative** 🡓 (instance does not occur), according the expected instances within the set period. If the expectation is met, then the prediction is **True**, otherwise it is **False**. Figure 10 exemplifies this classification.

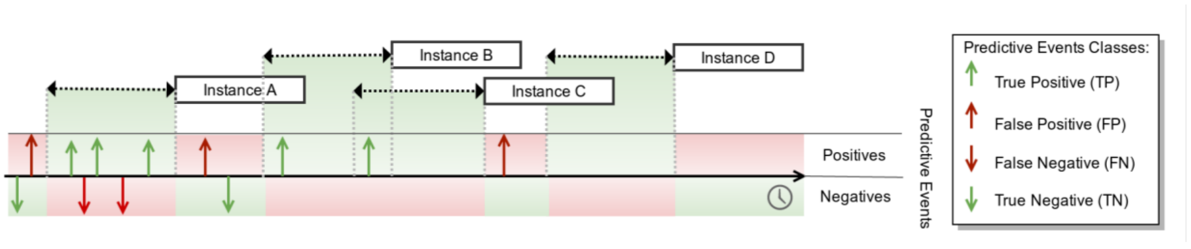


Figure 10 Simplified depictions of true/false positives/negatives

Consider a set (S) of instances of a target problem in a location, S = {A, B, C, D} → the set S contains all the instances of a target problem. Moreover, consider a set (TP) of instances of the set S, where TP Predictive events occur if TP = {A, B, C}. In addition, consider a set (FN) of instances of the set S, where FN Predictive events occur: FN = {A}. Lastly, consider a set (M) of instances of the set S, where there are no associated predictive events M = {D}.

With such definitions, the evaluations are defined as the following:

- Figure 11 plots the distribution of the number of associated TP predictions in the instances. → How many instances (axis Y) have a number of TP predictions in an instance (axis X, binned in intervals).

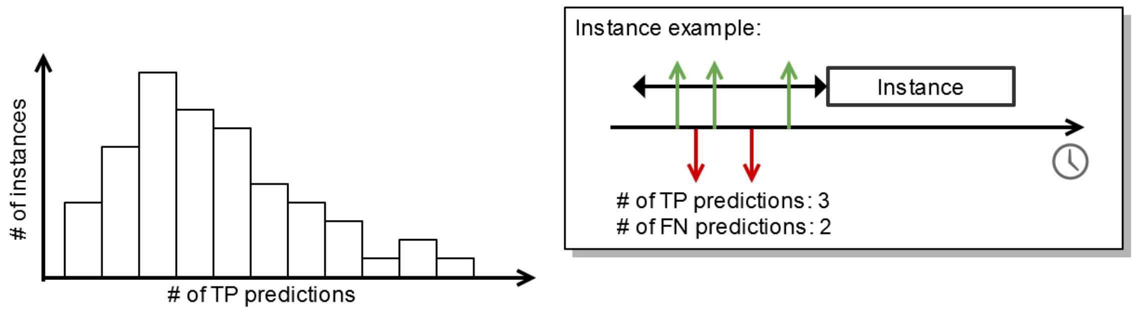


Figure 11 Distribution of the Number of TP Predictions in Instances

- Figure 12 plots the distribution of the maximum Δt in the instances. → How many instances (axis Y) have where predicted Δt before their start (axis X, binned in intervals.

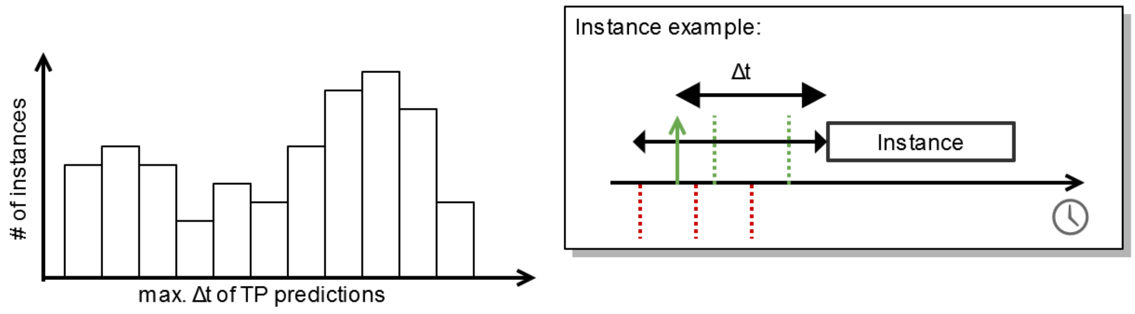


Figure 12 Distribution of the maximum Δt in the instances

- Figure 13 plots the distribution of the Δt in the associated TP predictions → How many TP predictions (axis Y) were made Δt before the start of their target instance (axis X, binned in intervals).

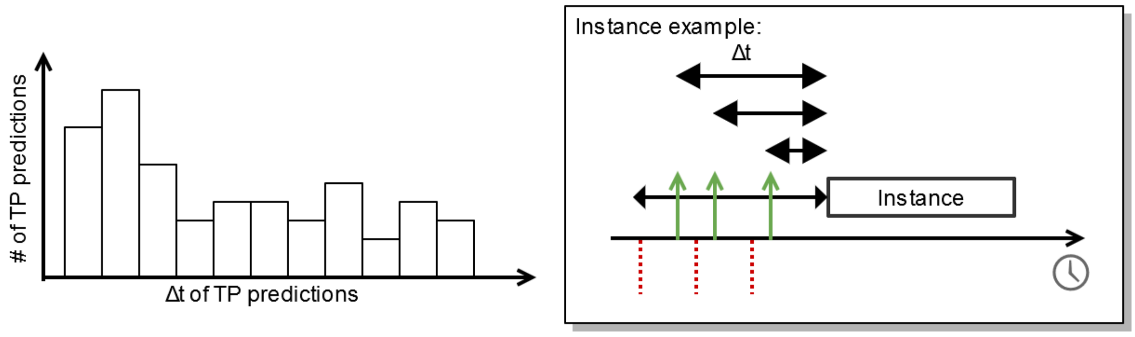


Figure 13 Distribution of the Δt in the associated TP predictions

For the set FN of the instances:

- Figure 14 plots the distribution of the Δt in the associated FN predictions → How many FN predictions (axis Y) were made Δt before the start of their target instance (axis X, binned in intervals).

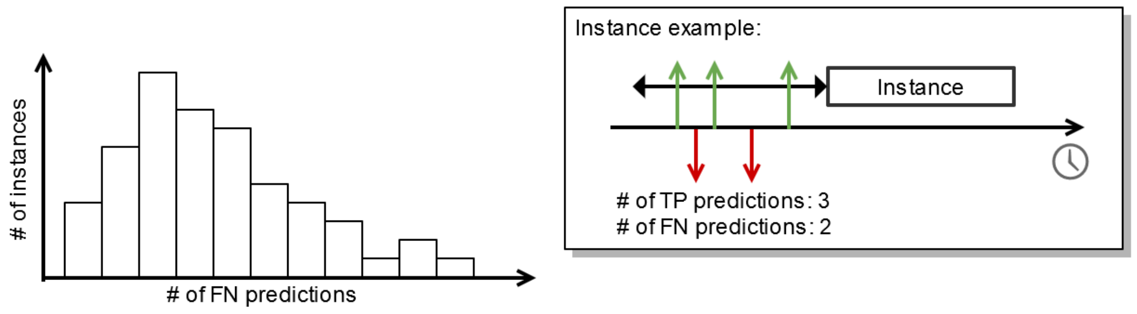


Figure 14 Distribution of the Δt in the associated FN predictions

For the set of predictions False Positive (FP):

- Figure 15 plots the distribution of the Δt in the FP predictions → How many FP predictions (axis Y) missed the target time by Δt (axis X, binned in intervals).

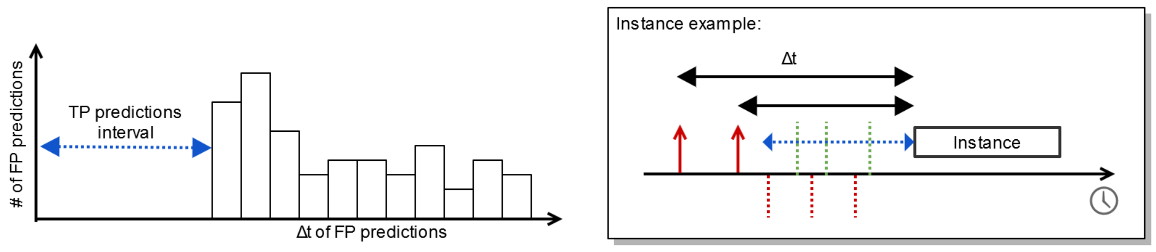


Figure 15 Distribution of the Δt in the FP predictions

3.1.2.1 Considerations

The predictive events that fall under the **False Positive** or **True Negative** classifications are not associated with an alarm instance. As such, it is not possible to evaluate them under the context of alarm instances. A predictive event can be associated with two or more target instances. The instances intervals can overlap as they come from different equipment within a location/slice. Instances that fall under the M set require a different Feature Engineering approach since there was no data retrieved with the current transformations.

3.1.3 Supervision Test Environment

Figure 16 depicts the processes deployed at AlticeLabs SG test lab to evaluate previously created models. The supervision processes are executed against historical data as they require the real events that match the predictions to evaluate the quality of the model results. In our test environment, the messages that arrive from the Kafka bus are persisted in a Hadoop storage to allow these validations.

When running the supervision processes, the end result is a report that provides the evaluation metrics and graphics for manual analysis. In future iterations of the supervision process, these metrics would serve as an input for an automated process that would switch the running predictive models by updated ones as needed or it would inform a human supervisor to take an action as the system degraded.

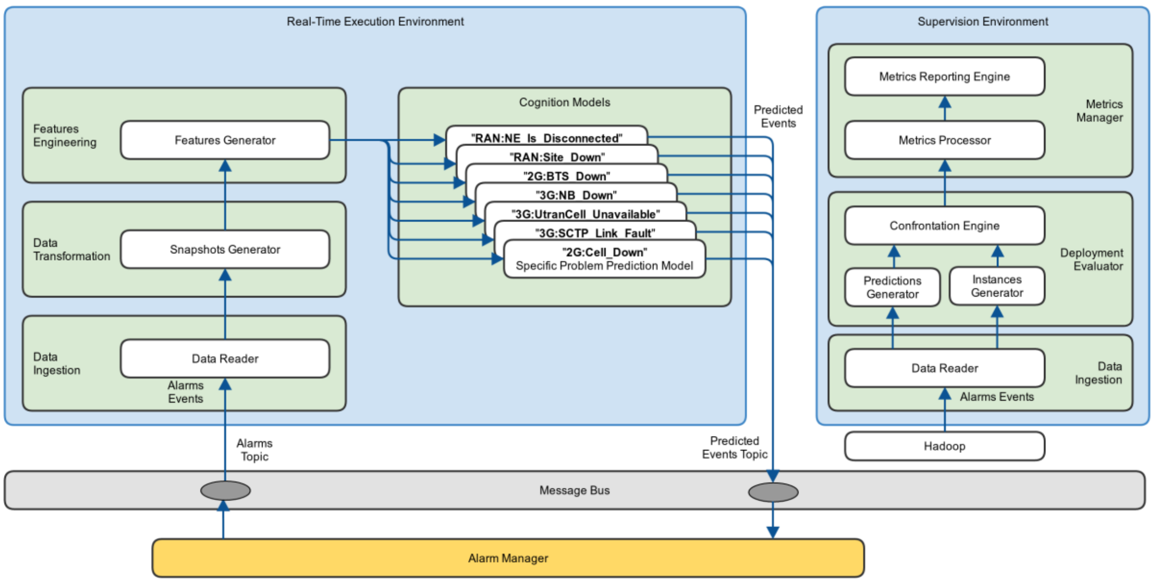


Figure 16 Execution Environment with Supervision

3.1.4 Results

The results obtained from the execution of the supervision processes are transcribed below in this section. For brevity reasons and since it would not provide much more information, the results here presented are just for a single day of execution of a single model.

Model: RAN:NE IS DISCONNECTED
Evaluation: 2019-08-08

Predicted **240** of measured **279** of **315** instances. **76.19%** of the instances were predicted successfully.

Events Metrics:

Event classification

	Real Negatives	Real Positives
Predicted Negatives	315474	858
Predicted Positives	2005	1038

Accuracy: 0.9910
Precision: 0.3411
Recall: 0.5475
F1: 0.4203

Figure 17 illustrates the FP predictive events to next instance occurrence (hours).

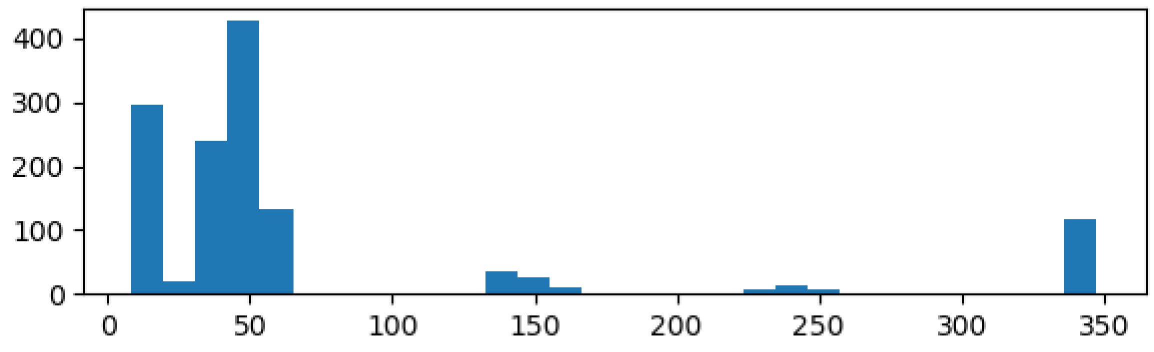


Figure 17 Distribution of FP Predictive Events to the next instance occurrence

Figure 18 depicts the instance metrics and model profiling.

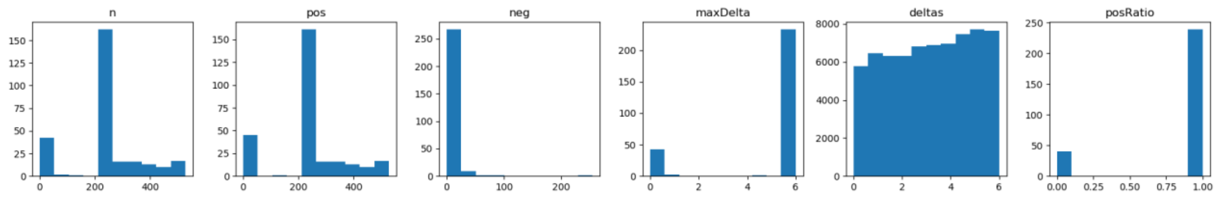


Figure 18 Instance Metrics and Model Profiling

In details, the depicted histograms have the following properties:

- n → x-axis describes the number of predictions for an instance in bins, the y-axis the number of instances.
- pos → x-axis describes the number of positive predictions for an instance in bins, the y-axis the number of instances.
- neg → x-axis describes the number of negative predictions for an instance in bins, the y-axis the number of instances.
- maxDelta → x-axis is the number of hours that the first positive prediction appears for an instance, y-axis is the number of instances.
- deltas → x-axis is the number of hours from the prediction to the instance, y-axis are the number of positive predictive events associated with an instance.
- PosRatio → x-axis describes the ratio of positive prediction by the number of predictions done for an instance, the y-axis the number of instances.

3.1.5 Final notes

Since the results were obtained for a test environment, no models were replaced by using these supervision processes. However, the developed tools allow understanding the performance of the deployed models that run in the Analyser. Without this development, the PF actuations could be missing or operating under wrong premises from the information given by the Analyser.

3.2 Noisy Neighbour (NN)

The Smart City UC is deployed as a NS composed by a set of network elements (VNFs), components and specific applications. This NS ensures in real time the connectivity of an enormous number of devices across the network (IoT devices). As such, the functions composing the NS are deployed in virtualized environments.

The developed ML model is used to detect if the perceived QoE is degraded due to the NN problem or not, as most elements of the network interconnecting the devices will be deployed in a virtualized environment. Since several machines may suffer from this problem in the infrastructure, an investigation should be tackled in order to determine the root cause of the noise for every deployed machine. Therefore, we propose a Root Cause Analysis (RCA) [4] approach in order to determine the source of the noise in the virtualized infrastructure. To achieve this purpose, we propose a propagation path-based solution that observes the correlation between events and seeks the most probable anomaly propagation sequence in the infrastructure. A graph is later created in order to describe the different dependencies in the system and the possible propagation paths of anomalies between the machines. Later, a score is defined to pinpoint the machine that is responsible for the noise phenomenon.

3.2.1 ML model

Propagation of anomalies between two machines in a virtualized infrastructure may occur following three scenarios as illustrated in Figure 19, where VM is a virtual machine and PM is a physical machine. VM to VM propagations are avoided as we believe propagation cannot occur unless it passes through PMs. Besides, on a large scale, it is very complicated to monitor every (VM-VM) couple in the infrastructure.

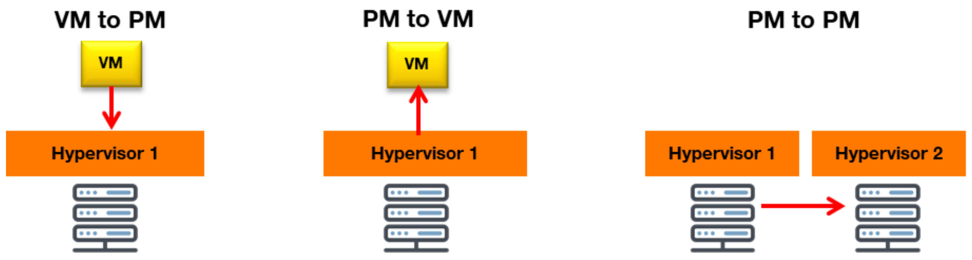


Figure 19 Noisy Neighbour propagation paths

The model aims to determine the propagation type of each couple of machines M1 and M2. Hence, each instance is described by six features: the CPU usage, the network inbound and the network outbound of each machine. The propagation path belongs to one of this set {PMtoVM, VMtoPM, PMtoPM}. In order to learn these propagation paths, we use a support vector machine (SVM) one-class classifier in order to train the propagation paths between each couple of machines. Different models per propagation path will decide the type of the propagation for every new data as illustrated in Figure 20.

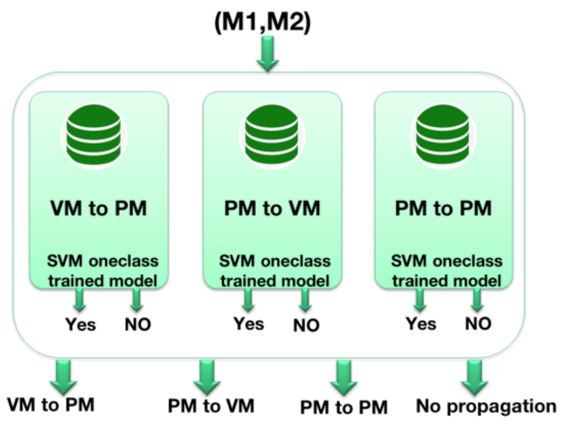


Figure 20 Noisy Neighbour ML classifier for propagation path determination

Once propagation paths are trained, the next step consists of creating a propagation graph that represents the correlation between abnormal system observations. Let G=(V,E) be a direct graph, where V is the set of misbehaving components and E represents the propagation paths.

In order to select the root cause, we assign an RCA score to each element of the graph. The RCA score represents the ability of the component to propagate its anomaly to the rest of the affected entities. It is the minimum distance that the entity needs in order to reach every component n in the system. The score of a component c in a graph G is the following: Score(c)$_G$= $\sum_{n \in G} d(c,n)$, where d(c,n) represents the shortest path distance from the component c to the component n using Dijkstra algorithm. If n is not accessible by c then $d_{\{c,n\}} = \infty$.

3.2.2 Experimental study

In order to evaluate the model, several experiments were executed in the NN testbed setup previously presented in D5.6 [2] as illustrated in Figure 21.

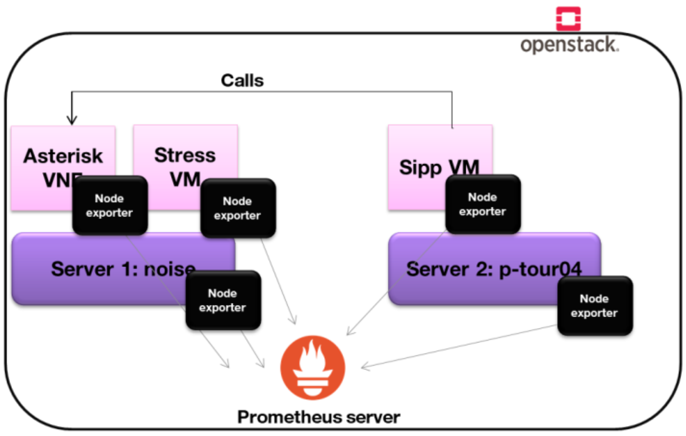


Figure 21 Noisy Neighbour Testbed Setup

"Noise" is a server containing the victim machine suffering from noise and the NN itself. "Asterisk" is an open source Voice over IP (VoIP) application and it represents our VNF deployed on a VM in order to receive calls. It is our victim. "Stress" is the NN VM that performs intense calculations and consumes a lot of CPU. "P-tour04" is another server on which we deploy a traffic generator "SipP". It is a free Open Source test tool for the Session Initiation Protocol (SIP). The role of this machine in the infrastructure is to generate calls to "Asterisk".

In the testbed, we use "Node Exporter" as an agent in every VM and PM. It allows collecting information from the machine in the form of counters that are sent to "Prometheus Server". This server creates metrics that describe the utilization of the component in order to create a dataset. The training data are created following different scenarios where components operate in different charges. Data are collected into "Prometheus" every 5 seconds, for about 8 days, for a total of 136920 instances per machine. 75% of the data are used for the training phase and the rest 25% are used for the test phase.

Once a NN is detected, an investigation is launched in the whole infrastructure in order to identify all the involved machines. The training data for RCA is collected from the couples of machines (Stress, Noise) labelled as {VMtoPM} and from the couple (Asterisk, Noise) labelled as {PMtoVM}. Once the SVM one-class model [5] is trained, the propagation paths between the machines are determined. The results over the test set are presented in Table 11. It shows a comparison of one-class SVM with a similar one-class algorithm, namely the Isolation forest [6]. The precision, recall and accuracy prove that the model succeeds to determine the propagation paths with an accuracy that is equal to 86%. Among all PM to VM propagations, 80% are detected. Besides, 78% of all VM to PM propagations are discovered by the model as well.

Table 11 Noisy Neighbour ML Results

Model (%)	one-class SVM		Isolation forest	
Accuracy (%)	86		81	
Propagation type	PMtoVM	VMtoPM	PMtoVM	VMtoPM
Precision (%)	100	100	100	100
Recall (%)	80	78	64	80
F1-score (%)	89	88	78	89

After determining the propagation paths, a direct graph is created as illustrated in Figure 22. This graph represents reachable components through propagation paths resulted from the previous step.

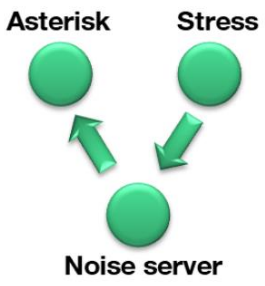


Figure 22 Noisy Neighbour propagation graph

Later, an RCA score is calculated in every node to describe the ability of that node to propagate its anomaly to the rest of the machines. According to the graph presented in Figure 22, we can see that both “Asterisk” machine and “Noise” server are unable to propagate the anomaly to the rest of machines. This later is proved by the score that has an infinite value for these two machines. However, “Stress” machine gives a finite score (score=3) since it can propagate its anomaly to the rest of the machines so it represents the root cause of our problem, it is then the NN. Hence, the migration of the VNF to another server can be performed after pinpointing the root cause of the problem thanks to this RCA solution.

3.3 QoE classification from QoS metrics

3.3.1 Goal

The goal of the QoS to QoE classification TUC is to use ML models to predict QoE at run-time and to trigger corrective measures within the SliceNet framework. The assumption is that the service provider can measure various QoS metrics; however, does not have full information on the actual QoE that the user is experiencing. Therefore, it must estimate the QoE from the measured QoS metrics. In particular, these QoE estimations would serve as triggers for actuations by the SliceNet QoE Optimizer. Given the monitored QoS parameters for the several deployed E2E NS, the corresponding QoE could be derived thanks to the proposed approach. Given such metric, a policy defined within the the SliceNet PF would define which actions (actuations) need to be triggered when facing bad QoE situations, for which the measurements would come as outputs from the proposed ML model. Then, these outputs would feed an instance of the QoE Optimizer, which will check the conditions stated by the slice policies in place and trigger the necessary (re-)configurations in case the conditions are met (e.g. the estimated QoE is below a certain threshold).

This approach can be exploited by the multiple SliceNet vertical UC, especially the eHealth one following the feedback approach stated previously, since it correlates the several network-level metrics (QoS) with the quality being perceived by the customers of the slice. Note that, while the developed ML may not match perfectly with the different vertical UCs expectations, mainly due to the different data employed in them, the methodology employed sets a solid framework to allow for QoE classification-based management of NSes within the context of the SliceNet architecture.

3.3.2 Scenario

The developed ML model focuses on the E2E latency of services over a network as distinctive QoE KPI. For this, a WordPress web service is employed [7], with the client and server running on two Kubernetes (K8s) clusters [8]. One K8s cluster corresponds to a managed slice, where a real slice provider would collect QoS metrics (e.g. through Skydive [9]) in order to manage the slice's desired QoE. In order to vary the WordPress client QoE, traffic application has been simulated with the iPerf3 tool [10], directing the traffic to the same host as the WordPress server. Then, the test driver collects and labels the QoE and the QoS metrics and uploads the metrics to a Jupyter notebook [11] for ML analysis. A ML classification process learns a QoE sensor model that estimates E2E QoE from measured QoS metrics. Finally, the model is then validated.

In order to simulate the user's workload, an application to generate WordPress client traffic to the WordPress server is run. The WordPress Client load is created with a Jmeter Load Tester tool [12]. The test driver utility is designed to coordinate the running of the WordPress client workloads, referred to as benchmark instances, concurrently with various levels of stress generated by iPerf3. The test driver also maintains an index that records the start time, end time, benchmark duration of each experimental sample used to map the benchmark duration (QoE) to the Skydive flows (QoS). The WordPress client benchmark consists of multiple concurrent get homepage, get posts, and downloads of files, 500KB, 5MB and 15MB, from the WordPress Server. Note that in D5.6 [2], the workload was simpler with only two threads of two downloads, 5MB and 15MB. In the background, various levels of stress are run, including no stress. Skydive is used as a sensor that collects per flow network metrics every minute from the WordPress server's eth0 interface (QoS). These one-minute segments are labelled with the benchmark instance id as they are collected. In D5.6, the sample size granularity was of an entire benchmark instance; now we partition the benchmark instances into minutes to mimic how the QoS will be collected in production.

To analyse the collected data, an IBM Watson Studio [13] notebook with a Python 3.6 kernel is employed, with the Python Data Analysis Library (Pandas [14]) used to aggregate the QoS and QoE measures. Finally, the Python Scikit-learn library [15] is used for creating the ML model.

3.3.3 Data Description

3.3.3.1 ML Features

The different per flow metrics collected through Skydive are transformed and aggregated to supply the QoS features used to generate the ML model for predicting QoE classifications, namely binary and multi-class QoE classification of benchmarks durations. We start with the raw Transmission Control Protocol (TCP) flow metrics described in Table 12, where A and B are the source and destination Internet Protocol (IP) addresses of a flow, respectively. In D5.6, it was also used these raw TCP flow metrics for its evaluation.

Table 12 Skydive raw TCP network flow metrics

Skydive raw TCP network flow metrics	Description
Metric.Start	Flow start time
Metric.Last	Flow end time
Metric.ABBytes	Number of bytes sent from A to B
Metric.BABytes	Number of bytes sent from B to A
Metric.ABPackets	Number of data packets sent from A to B
Metric.BAPackets	Number of data packets sent from B to A
Metric.RTT	Round Trip Time

These raw flow metrics described in Table 12 are transformed into per flow metrics described in Table 13.

Table 13 Skydive per flow transformations

Skydive per flow transformations	Description
flow_duration	Metric.Last - Metric.Start
bytes_per_flow	(Metric.ABBytes + Metric.BABytes) / flow_duration
packets_per_flow	(Metric.ABPackets + Metric.BAPackets) / flow_duration
AB_bytes_per_flow	Metric.ABBytes / flow_duration
BA_bytes_per_flow	Metric.BABytes / flow_duration
AB_packets_per_flow	Metric.ABPackets / flow_duration
BA_packets_per_flow	Metric.BAPackets / flow_duration
Metric.RTT	Round Trip Time

The per flow transformations above are aggregated in to per minute aggregations described in Table 14. In D5.6, the aggregations were per benchmark instance not time windows.

Table 14 Skydive per minute aggregations

Skydive per minute aggregations	Description
flow_duration_mean	Flow_duration per minute mean
bytes_per_flow_mean	Mean of bytes per flow_duration per minute
packets_per_flow_mean	Mean of packets per flow_duration per minute
AB_bytes_per_flow_mean	Mean of AB bytes per flow_duration per minute
BA_bytes_per_flow_mean	Mean of BA bytes per flow_duration per minute
AB_packets_per_flow_mean	Mean of AB packets per flow_duration per minute
BA_packets_per_flow_mean	Mean of BA packets per flow_duration per minute
RTT_mean	RTT per flow per minute mean
flow_duration_count	Number flows per minute
flow_duration_sum	Sum of flow_durations per minute
bytes_per_flow_sum	Sum of bytes per flow_duration per minute
packets_per_flow_sum	Sum of packets per flow_duration per minute
AB_bytes_per_flow_sum	Sum of AB bytes per flow_duration per minute
BA_bytes_per_flow_sum	Sum of BA bytes per flow_duration per minute
AB_packets_per_flow_sum	Sum of AB packets per flow_duration per minute
BA_packets_per_flow_sum	Sum of BA packets per flow_duration per minute
RTT_sum	Sum of RTTs per minute
bytes_sum	Bytes per minute
packets_sum	Packets per minute
AB_bytes_sum	AB bytes per minute
BA_bytes_sum	BA bytes per minute
AB_packets_sum	AB packets per minute
BA_packets_sum	BA packets per minute

Table 15 describes the correlations between the QoS features described in Table 14 with the target QoE, namely benchmark duration.

Table 15 QoS/QoE Correlations

Skydive per minute aggregations (QoS)	Pearson Correlation to Benchmark duration (QoE)
flow_duration_count	0.138526
flow_duration_sum	0.005999
RTT_mean	-0.019630
flow_duration_mean	-0.020285
RTT_sum	-0.026374
AB_bytes_per_flow_mean	-0.355992
AB_bytes_per_flow_sum	-0.439918
AB_packets_per_flow_mean	-0.457886
AB_packets_per_flow_sum	-0.538687
BA_packets_per_flow_mean	-0.587640
BAPackets_sum	-0.587990
packets_sum	-0.589249
packets_per_flow_mean	-0.590046
BABytes_sum	-0.591192
bytes_sum	-0.591318
ABPackets_sum	-0.593416
BA_bytes_per_flow_mean	-0.600813
bytes_per_flow_mean	-0.604522
ABBytes_sum	-0.611438
packets_per_flow_sum	-0.694955
BA_packets_per_flow_sum	-0.703995
BA_bytes_per_flow_sum	-0.715410
bytes_per_flow_sum	-0.718420

The QoS features with high negative correlation (coloured in green) are used in generating the ML classification models to predict QoE from QoS. Specifically, we use features: ***packets_per_flow_sum***, ***BA_packets_per_flow_sum***, ***BA_bytes_per_flow_sum***, and ***bytes_per_flow_sum***. In D5.6, no such correlation or feature prioritization was done.

3.3.3.2 ML Target Classifications

The ML target classifications represent the target QoE and are based on benchmark durations with no background stress. Table 16 describes the QoE Classifications and decision boundaries. The boundaries are derived by calculating quantiles over the benchmark duration instances with no background stress. The high boundary is the .99 quantile and the low boundary .75 quantile.

Table 16 QoE Classifications and decision boundaries.

Classification Type	QoE Classification	Class	Formula	Range
Binary	Good	0	range(0,quantile(0.99))	0-812559
Binary	Bad	1	range(quantile(0.99),infinity)	812559-infinity
Multiclass	Good	0	range(0,quantile(0.75))	0-482081
Multiclass	Acceptable	1	range(quantile(0.75),quantile(0.99))	482081-812559
Multiclass	Bad	2	range(quantile(0.99),infinity)	812559-infinity

Table 17 describes the QoE Classification training and test data distribution over the QoE classifications described in Table 16. The source of the QoE data for the training and test data is real data calculated for each benchmark instance’s duration. The training dataset includes 530 benchmark instances divided into **4845** one-minute segments of measured QoS flow metrics, while the testing set includes 198 benchmark instances divided into **1901** one-minute segments of measured QoS flow metrics. The Training/Testing one-minute segments of measured labelled QoS data labelled with the associated benchmark instance id.

Table 17 QoE from QoS ML Classification Training and Test Data Distribution

Classification Type	QoE Classification	Sample set	Benchmark instances	One Minute Segments
Binary	Good	Training	377	**2850**
Binary	Bad	Training	153	**1995**
Binary	Good	Testing	106	**701**
Binary	Bad	Testing	92	**1200**
Multiclass	Good	Training	243	**1545**
Multiclass	Acceptable	Training	134	**1305**
Multiclass	Bad	Training	153	**1995**
Multiclass	Good	Testing	86	**535**
Multiclass	Acceptable	Testing	20	**166**
Multiclass	Bad	Testing	92	**1200**

3.3.4 Results

We used a few methods to predict QoE from QoS. The methods are summarized in Table 18. In order to determine which method performed best we rank their predictions by their F1, accuracy, and log loss scores. The results are described in the following sections. In D5.6, only the Non-Neural ML Classifier Basket was employed so there was not method comparison.

Table 18 QoE from QoS Prediction Methods

QoE from QoS Prediction Methods	Description
Thresholding	Thresholding uses classic analytics to predict QoE from a single high correlation QoS feature without resorting to any of the available ML methodologies.
Neural Network	MLPClassifer is scikit's neural network classifier. MLPClassifer takes few hours to run so it was only run on the full complement of features that exhibit a high correlation with the target QoE.
TPOT	TPOT is a Python Automated ML tool that optimizes ML pipelines using genetic programming. TPOT explores thousands of possible pipelines to find the best one for your data. It takes days to run so it was only run on the full complement of features that exhibit a high correlation with the target QoE.
Non-Neural ML Classifier Basket	A basket of Non-Neural ML Classifiers is used to best predict the target QoE. Each classifier is exercised against all combinations of the features with high correlation to the target QoS.

3.3.4.1 Thresholding

Thresholding uses classic analytic techniques to predict QoE from a single high correlation QoS feature without using a ML methodology. It assumes that since the subject QoS feature has a high correlation with target QoE, its distribution will also be the same, so it mimics the same boundary conditions used to define the QoE classes, but it applies them to the QoS feature. Specifically, the boundaries are derived by calculating quantiles over the subject QoS feature per minute samples with no background stress. The high boundary is the .99 quantile and the low boundary .75 quantile. Table 19 and

Table *20* show the resulting threshold boundary ranges for QoS features **bytes_per_flow_sum** and **bytes_sum** per minute aggregations. Note that since these feature exhibit high negative correlation with the target QoE their classifications are inverted. That is, the target QoE, i.e. benchmark duration, is better when it less, whereas the QoS feature, i.e. bytes per minute, is better when it is greater.

Table 19 QoS Threshold boundary ranges for bytes_per_flow_sum per minute.

Classification Type	QoS Threshold Classification	Class	Formula	Range
Binary	Bad	1	range(0,quantile(0.99))	0-16619
Binary	Good	0	range(quantile(0.99),infinity)	16619-infinity
Multiclass	Bad	2	range(0,quantile(0.75))	0-11328
Multiclass	Acceptable	1	range(quantile(0.75),quantile(0.99))	11328-16619
Multiclass	Good	0	range(quantile(0.99),infinity)	16619-infinity

Table 20 QoS Threshold boundary ranges for bytes_sum per minute.

Classification Type	QoS Threshold Classification	Class	Formula	Range
Binary	Bad	1	range(0,quantile(0.99))	0-3432942
Binary	Good	0	range(quantile(0.99),infinity)	3432942-infinity
Multiclass	Bad	2	range(0,quantile(0.75))	0-221748999
Multiclass	Acceptable	1	range(quantile(0.75),quantile(0.99))	221748999-3432942
Multiclass	Good	0	range(quantile(0.99),infinity)	3432942-infinity

The prediction results are described in Table 21 and their respective confusion matrixes are described in Table 22 and Table 23. Predictions with ***bytes_per_flow_sum*** were better than ***bytes_sum***, but neither did a very good job.

Table 21 QoE from QoS Predictions with Thresholding Results

Classification	Classifier	Features (per minute aggregations)	F1 score	Accuracy score
Binary	Custom made	bytes_per_flow_sum	0.64	0.64
Multiclass	Custom made	bytes_per_flow_sum	0.65	0.65
Binary	Custom made	bytes_sum	0.63	0.63
Multiclass	Custom made	bytes_sum	0.28	0.28

Table 22 QoE from QoS Predictions with Thresholding on bytes_per_flow_sum per minute Confusion Matrixes

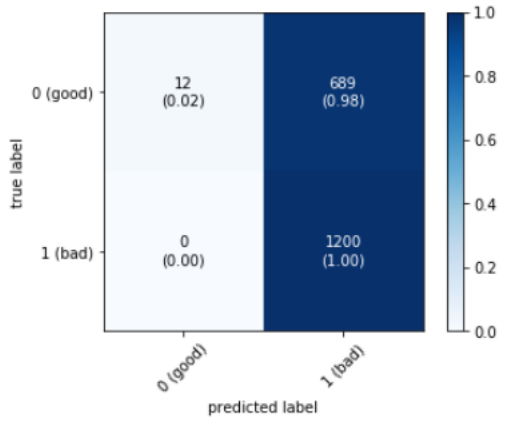
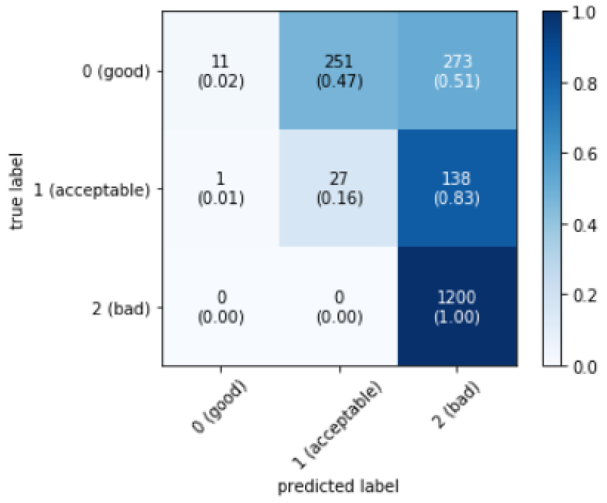
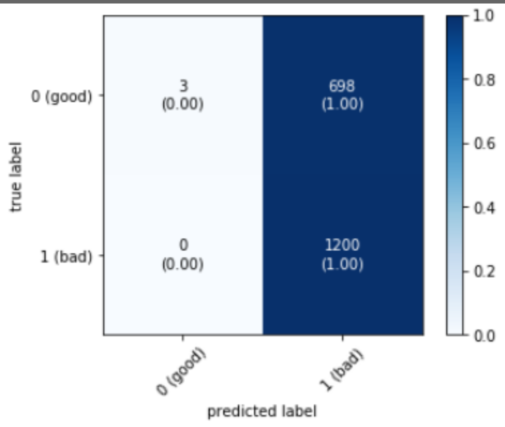
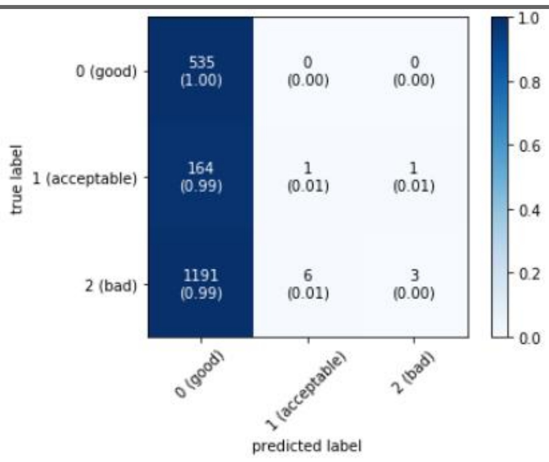
Binary Classification Confusion Matrix	**Multiclass Classification Confusion Matrix**
	

Table 23 QoE from QoS Predictions with Thresholding on bytes_sum per minute Confusion Matrixes

Binary Classification Confusion Matrix	**Multiclass Classification Confusion Matrix**
	

3.3.4.2 Neural Network

MLPClassifer is scikit's neural network classifier [16]. It is a Multi-Layer Perceptron (MLP). We ran it using parameter settings of solver='adam', alpha=1, max_iter=1000, hidden_layer_sizes=(1000,1000,1000). We also tried solvers lbfgs and sgd, but the solver adam performed better so only its results are presented below. For multiclass we set the resulting model to 'softmax' before doing the prediction (as recommended in its documentation). MLPClassifer (with deep hidden_layers) takes a few hours to run so it was only run on the full complement of the features that highly correlate with target QoE (not all combinations of these same features) described in Table 15 above.

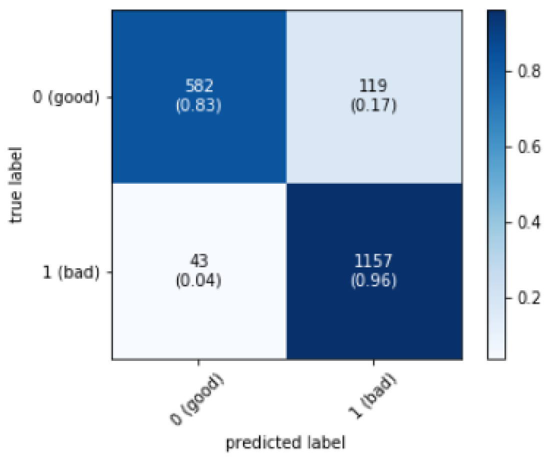
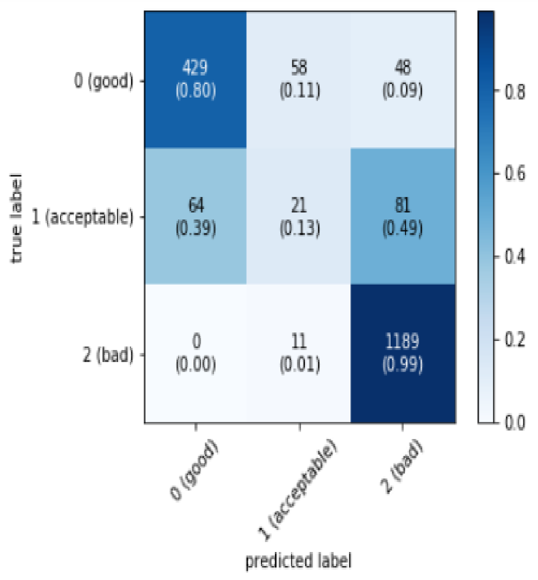
The results are described in Table 24,

Table 25, Figure 23, and Figure 24. Its predictions for Binary Classification returns F1 and accuracy scores in the low .90s, while its predictions for Multiclass are less promising with F1 and accuracy scores in the mid .80s.

Table 24 QoE from QoS Predictions with MLPClassifier Results

Classification	**Classifier**	**Features**	**F1 score**	**Accuracy score**	**Log loss score**
Binary	MLPClassifier	packets_per_flow_sum, BA_packets_per_flow_sum, BA_bytes_per_flow_sum, bytes_per_flow_sum	0.92	0.92	0.38
Multiclass	MLPClassifier	packets_per_flow_sum, BA_packets_per_flow_sum, BA_bytes_per_flow_sum, bytes_per_flow_sum	0.86	0.86	0.60

Table 25 QoE from QoS Predictions with MLPClassifier Confusion Matrixes

Binary Classification	**Multiclass Classification**
	

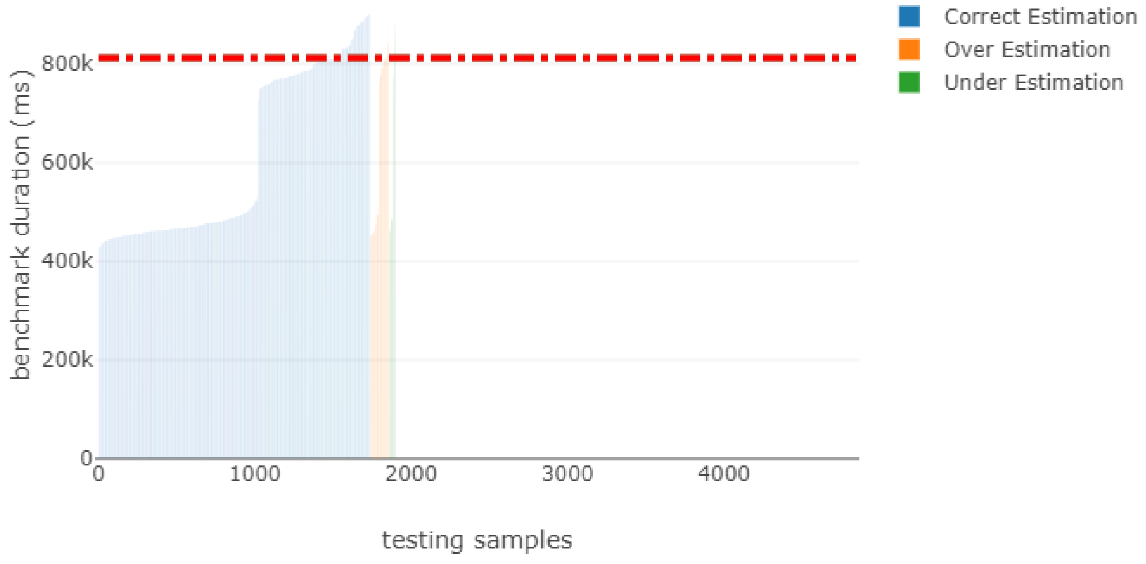


Figure 23 QoE from QoS Predictions with MLPClassifier for Binary Classification Distribution

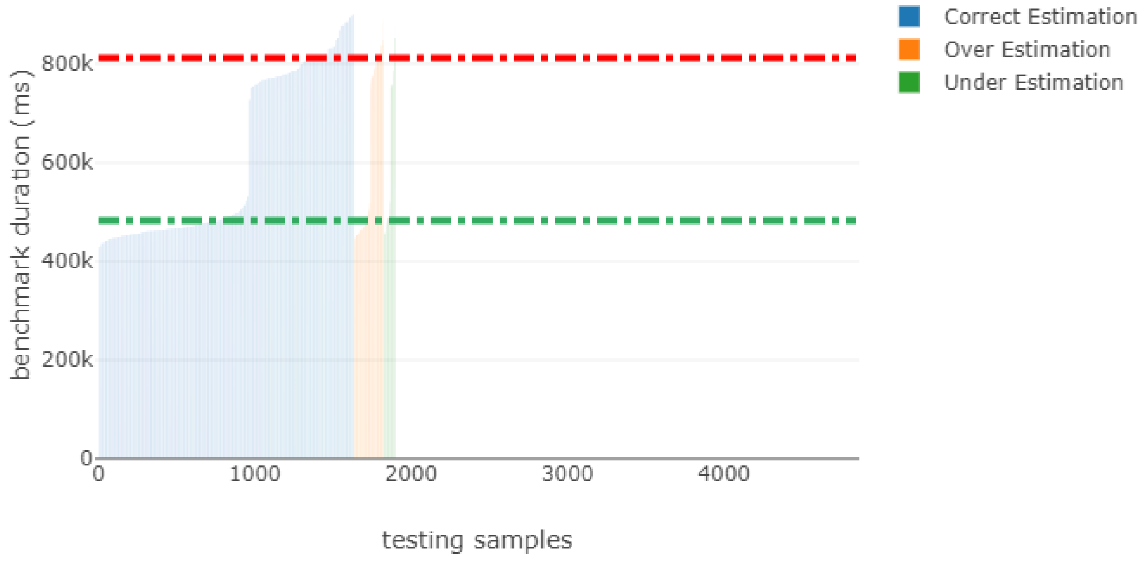


Figure 24 QoE from QoS Predictions with MLPClassifier for Multiclass Classification Distribution

3.3.4.3 TPOT

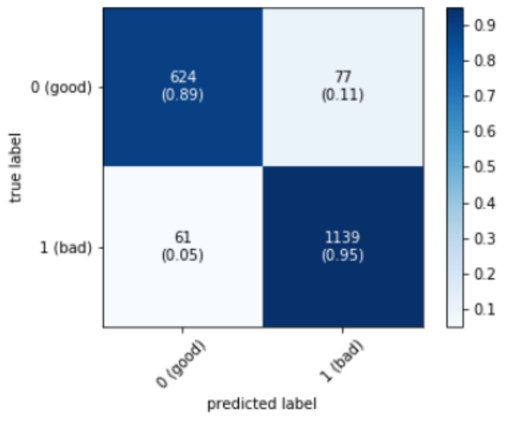
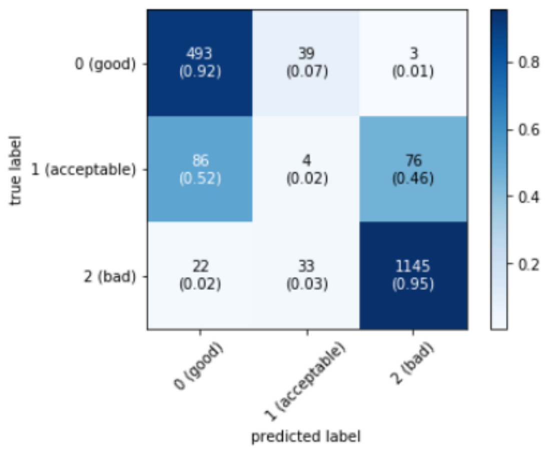
TPOT [17] is a Python Automated ML tool that optimizes ML pipelines using genetic programming. TPOT explores thousands of possible pipelines to find the best one for the subject input data. It takes days to run so it was only run on the full complement of the features that highly correlate with target QoE (not all combinations of these same features) described in Table 15 above.

The results are described in Table 26, Table 27, Figure 25, and Figure 26. Its prediction for Binary Classification returns F1 and accuracy scores in the low .90s, while its predictions for Multiclass are less promising with F1 and accuracy scores in the mid .80s.

Table 26 QoE from QoS Predictions with TPOT Results

Classification	**Classifier**	**Features**	**F1 score**	**Accuracy score**	**Log loss score**
Binary	RandomForestClassifier	packets_per_flow_sum, BA_packets_per_flow_sum, BA_bytes_per_flow_sum, bytes_per_flow_sum	0.93	0.92	0.26
Multiclass	ExtraTreesClassifier	packets_per_flow_sum, BA_packets_per_flow_sum, BA_bytes_per_flow_sum, bytes_per_flow_sum	0.86	0.86	0.41

Table 27 QoE from QoS Predictions with TPOT Confusion Matrixes

Binary Classification	**Multiclass Classification**
	

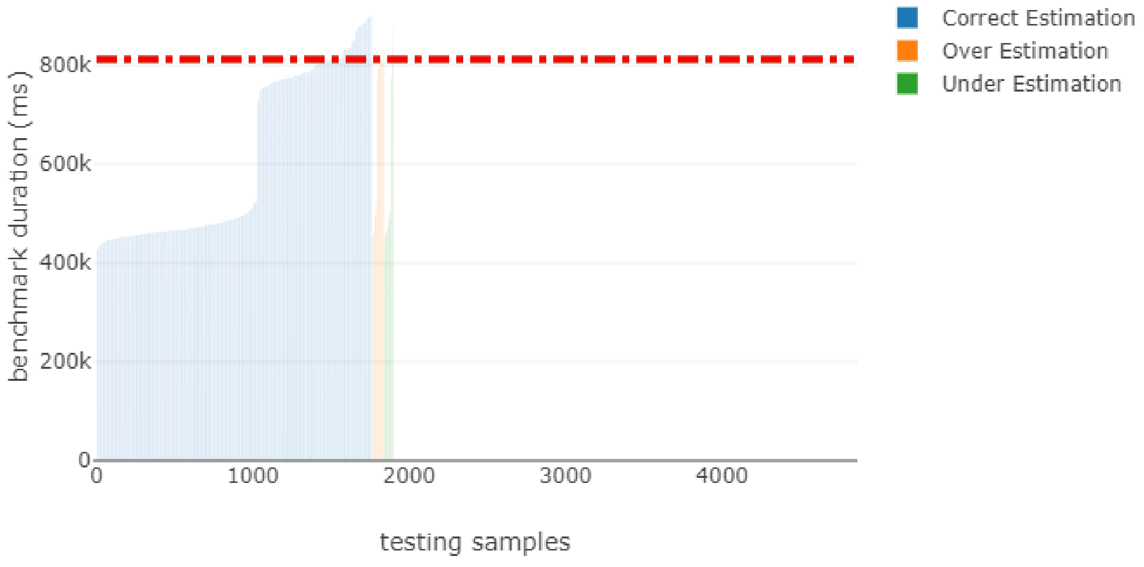


Figure 25 QoE from QoS Predictions with TPOT for Binary Classification Distribution

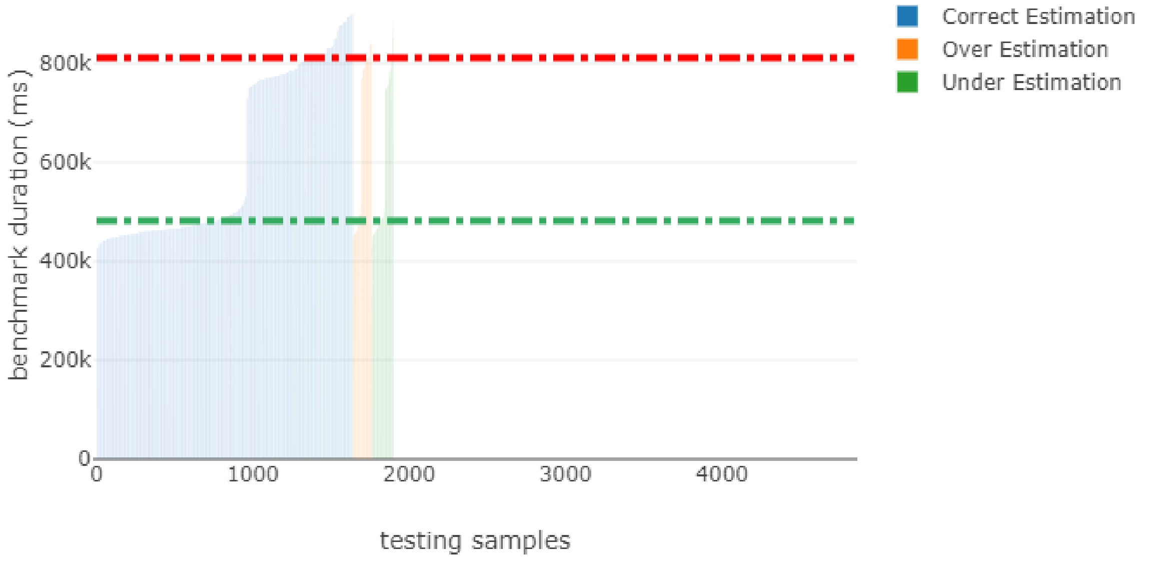


Figure 26 QoE from QoS Predictions with TPOT for Multiclass Classification Distribution

3.3.4.4 Non-Neural ML Classifiers Basket

A basket of Non-Neural ML Classifiers described in Table 28 are used in the analysis to best predict the target QoE. Each classifier is exercised against all combinations of the high (negative) correlation features described in Table 15. The classifiers were compared to determine which best predicts the target QoE using the selected QoS features. In order to determine which classifier and feature combinations perform best we rank their predictions by their F1, accuracy, and log loss scores.

Table 28 QoE from QoS Non-Neural Basket of ML Classifiers

Classifiers	Description
LogisticRegression	Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function [18].
DecisionTreeClassifier	Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [19].
KNeighborsClassifier	Classifier implementing the k-nearest neighbours vote [20].
LinearDiscriminantAnalysis	A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule. The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix. The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions [21].
RandomForestClassifier	A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default) [22].
GaussianNB	Gaussian Naive Bayes (GaussianNB) [23].

SVC	The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples [24].
AdaBoostClassifier	An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. This class implements the algorithm known as AdaBoost-SAMME [25].
QuadraticDiscriminantAnalysis	A classifier with a quadratic decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule. The model fits a Gaussian density to each class [26].

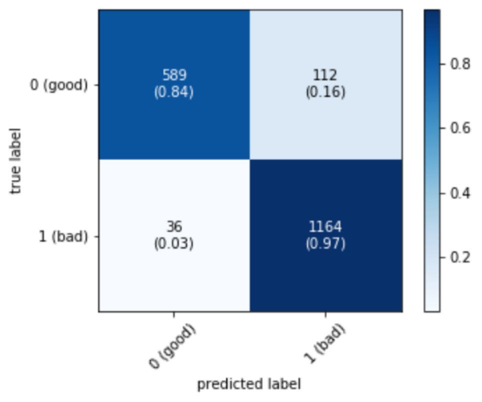
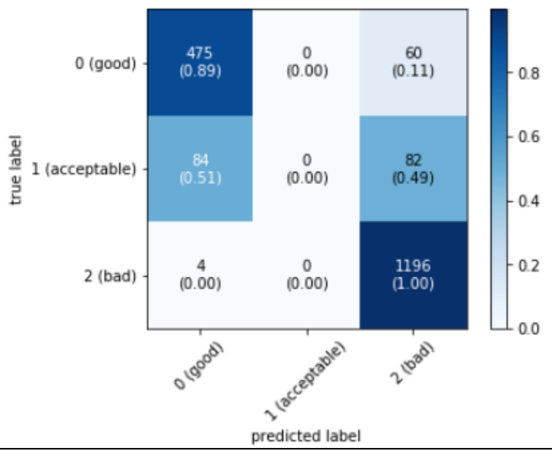
The results are described in Table 29,

Table *30*, Figure 27 and Figure 28. Its prediction for Binary Classification returns F1 and accuracy scores in the low .90s, while its predictions for Multiclass returns a F1 score in the low .90s and an accuracy score in the high .80s. A closer look at the Multiclass confusion matrix shows that it did not predict any of the middle classification, i.e. **1** (**acceptable**), correctly. It either under estimated or over estimated.

Table 29 QoE from QoS Predictions with Non-Neural ML Classifiers Basket Results

Classification	**Classifier**	**Features**	**F1 score**	**Accuracy score**	**Log loss score**
Binary	AdaBoostClassifier	packets_per_flow_sum, BA_packets_per_flow_sum, BA_bytes_per_flow_sum	0.92	0.92	0.65
Multiclass	LogisticRegression	packets_per_flow_sum, BA_packets_per_flow_sum, bytes_per_flow_sum	0.92	0.88	0.70

Table 30 QoE from QoS Predictions with Non-Neural ML Classifiers Basket Confusion Matrixes

Binary Classification	**Multiclass Classification**
	

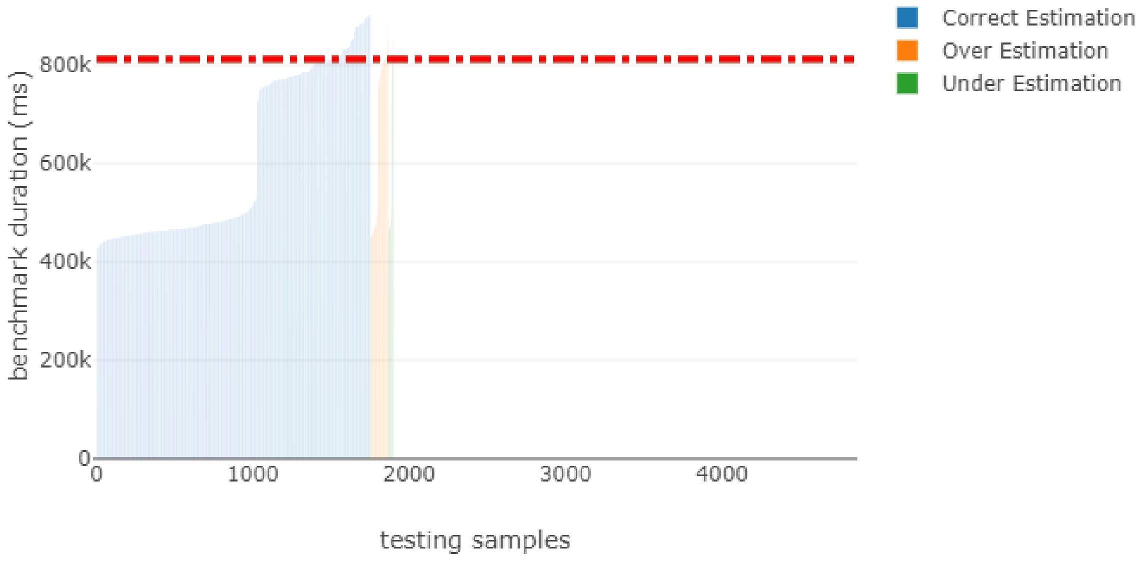


Figure 27 QoE from QoS Predictions with Non-Neural ML Classifiers Basket for Binary Classification Distribution

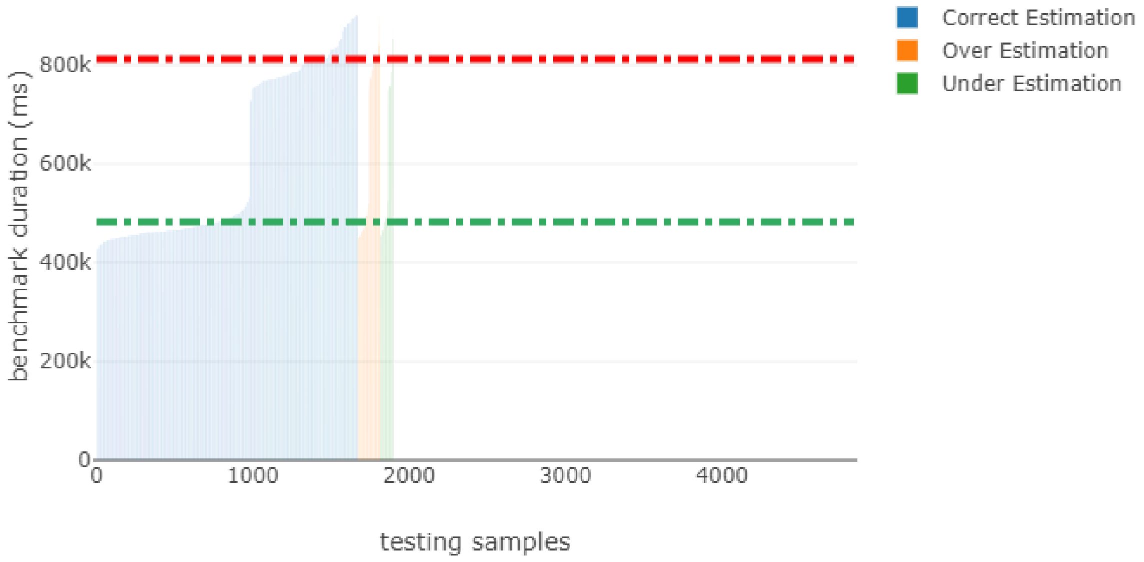


Figure 28 QoE from QoS Predictions with Non-Neural ML Classifiers Basket for Multiclass Classification Distribution

3.3.4.5 Conclusion

Overall the Non-Neural Basket of ML Classifiers method out performs the other classification methods. Its predictions are better, and it takes less time to create a model than the other ML methods. We think its results could be improved by tweaking the input parameters the classifiers.

In a production setting where the classifier predictions could be used to stimulate some remedial action, incorrect predictions could have negative consequences. Over estimation could result in allocation of more additional resources than required. Whereas, under estimation could result in inaction, whose consequent might be user dissatisfaction or an SLA violation. Hence, the main goal is to use the developed ML models to predict QoE at run-time and to trigger corrective measures within the SliceNet framework. In particular, QoE estimations would serve as triggers for actuations by the SliceNet QoE Optimizer. Based on the monitored QoS parameters for the several deployed E2E NS (fed into the DSP's Data Lake), the corresponding QoE classification (hosted at the Analyser) could be derived from the ML model generated during the train phase. On the basis of the QoE classification and other QoS metrics, policies would be defined within SliceNet PF that would trigger actions to remedy unsatisfactory QoE levels. This approach can be exploited by the multiple SliceNet vertical UCs, especially the eHealth one, since it correlates the several network-level metrics (QoS) with the quality being perceived by the customers of the NS.

3.4 RAN Optimisation

We analyse the RAN status, produced through the Mosaci5G FlexRAN platform, in order to create a regression model using the correlation and relation of different metrics with each other. The main purpose of the model is to predict the user channel quality indicators (CQI), denoted as wide-band CQI (wbCqi), with given important statistics. A properly trained wbCQI based model with high accuracy can be used for many applications such as:

- Predicting the wbCQI in the near future
    - Predicting possible UE connection loss in advance and taking actions beforehand.
    - Calculate velocity and mobility of the UE from the spatio-temporal variability.
- Predicting base station status and achievable rate
    - Mal-function devices or coverage issue in particular geographical areas.

In particular, and using the SliceNet's Smart City UC as an example, the wbCQI can be used to determine connectivity of massive number of devices and identify the mal-function devices or geographical areas. As our initial approach for RAN optimisation, the channel quality and therefore the achievable rate of a given UE will be derived from its wbCQI. It has to be noted that any other metric can be used in the future to create more complex models to predict the desired behaviour in different situations. In this context, we applied the following steps as a methodology to derive our model:

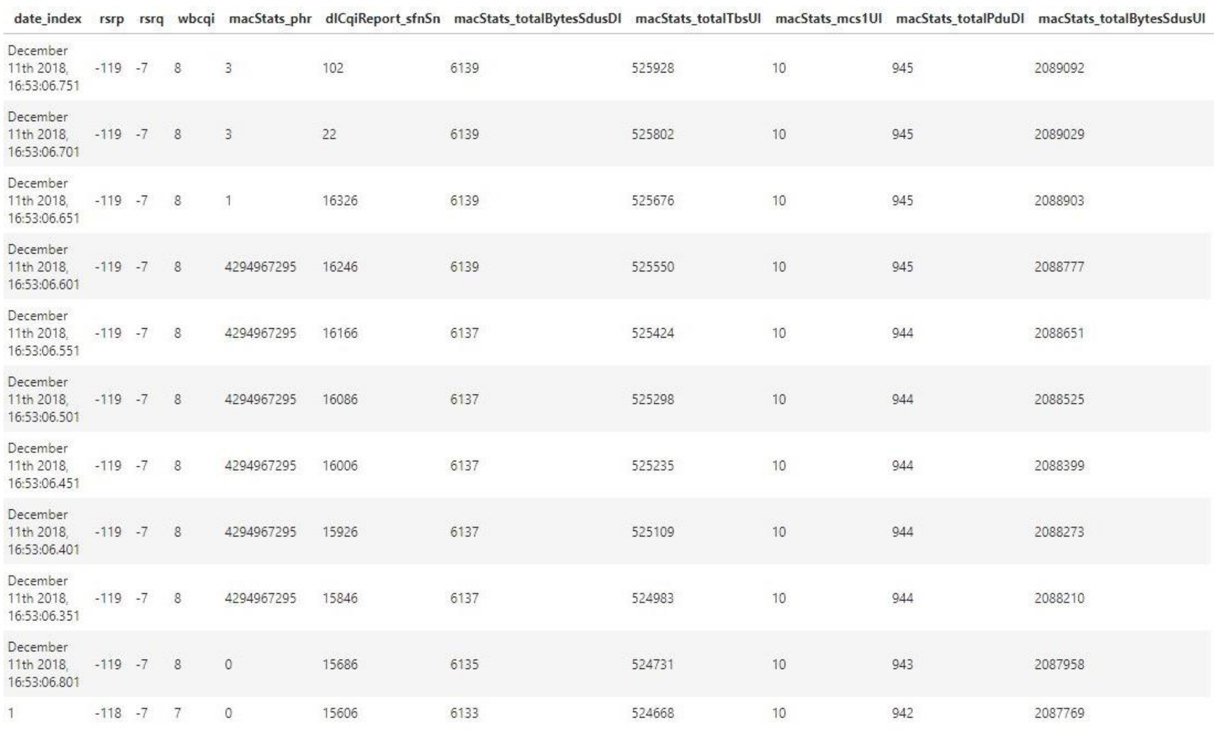
1. Data cleaning (sanitization).
2. Data pre-processing.
3. Data analysis.
4. Regression model training.
5. Save/Load of trained models.
6. Validate and Evaluate results.
7. Generate real time predictor.

Multiple RAN datasets have been generated at EURECOM and contributed to Crawdad [27].

3.4.1 Data Cleaning and Pre-processing

As stated above, Mosaic5G FlexRAN platform is used to produce datasets for RAN optimisation in a Java Script Object Notation (JSON) format with more than 100 metrics per measurement. In order to monitor and train on data in a given time and space, timestamp metadata is added for each measured data sample inside the JSON tree to identify the time elapsed between different measurements. In addition, some of these metrics remain unchanged regardless of user mobility, which is why we make use of a second order data pre-processing to remove those metrics that do not change over time (see Table 31).

Table 31 Second order RAN data pre-processing sample



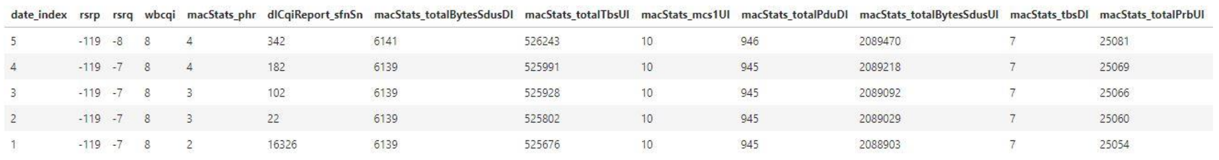
date_index	rsrp	rsrq	wbcqi	macStats_phr	dlCqiReport_sfnSn	macStats_totalBytesSdusDl	macStats_totalTbsUl	macStats_mcs1Ul	macStats_totalPduDl	macStats_totalBytesSdusUl
December 11th 2018, 16:53:06.751	-119	-7	8	3	102	6139	525928	10	945	2089092
December 11th 2018, 16:53:06.701	-119	-7	8	3	22	6139	525802	10	945	2089029
December 11th 2018, 16:53:06.651	-119	-7	8	1	16326	6139	525676	10	945	2088903
December 11th 2018, 16:53:06.601	-119	-7	8	4294967295	16246	6139	525550	10	945	2088777
December 11th 2018, 16:53:06.551	-119	-7	8	4294967295	16166	6137	525424	10	944	2088651
December 11th 2018, 16:53:06.501	-119	-7	8	4294967295	16086	6137	525298	10	944	2088525
December 11th 2018, 16:53:06.451	-119	-7	8	4294967295	16006	6137	525235	10	944	2088399
December 11th 2018, 16:53:06.401	-119	-7	8	4294967295	15926	6137	525109	10	944	2088273
December 11th 2018, 16:53:06.351	-119	-7	8	4294967295	15846	6137	524983	10	944	2088210
December 11th 2018, 16:53:06.801	-119	-7	8	0	15686	6135	524731	10	943	2087958
1	-118	-7	7	0	15606	6133	524668	10	942	2087769

Even after removing (quasi-)constant metrics and keeping only the relevant and dynamic data across 42 columns, it is noticed that some integer overflow occurred, e.g. on *macStats_phr,* during the recording, which are resolved during cleaning and sanitizing the datasets. As for the cleaning steps, we perform the following two operations, with Table 32 depicting an example of the cleaned dataset:

- Aggregating of multiple columns and using the mean or median of other columns to remove overflows and sudden metric spike,
- Changing date time format showing it is growing to future rather than exact dates (exact dates give no useful information).

At the end of this step, the pre-processing of the dataset is completed. A total of 42 fields are left that might or might not be used for the regression model. The recordings contain no bad data and data index problem is resolved after this step. Following box illustrates the list of 42 features left in the dataset for the further analysis part.

Table 32 Cleaned RAN dataset sample



date_index	rsrp	rsrq	wbcqi	macStats_phr	dlCqiReport_sfnSn	macStats_totalBytesSdusDl	macStats_totalTbsUl	macStats_mcs1Ul	macStats_totalPduDl	macStats_totalBytesSdusUl	macStats_tbsDl	macStats_totalPrbUl
5	-119	-8	8	4	342	6141	526243	10	946	2089470	7	25081
4	-119	-7	8	4	182	6139	525991	10	945	2089218	7	25069
3	-119	-7	8	3	102	6139	525928	10	945	2089092	7	25066
2	-119	-7	8	3	22	6139	525802	10	945	2089029	7	25060
1	-119	-7	8	2	16326	6139	525676	10	945	2088903	7	25054

3.4.2 Data Analysis

We have recorded different patterns from different scenarios of UE moving away or closer with respect to the eNB or remaining at the same position without any mobility. Based on these datasets, we generated the correlation matrix to identify metric dependencies.

3.4.2.1 Correlation Matrix

We created a correlation matrix from fields listed in the list below to reveal the correlation level of different features in a dataset against the other fields (DataFrame Corr is used for this purpose). This is an important factor as it provides a list of useful and not useful fields. *['date_index', 'rsrp', 'rsrq', 'wbcqi', 'macStats_phr', 'dlCqiReport_sfnSn', 'macStats_totalBytesSdusDl', 'macStats_totalTbsUl', 'macStats_mcs1Ul', 'macStats_totalPduDl', 'macStats_totalBytesSdusUl', 'macStats_tbsDl', 'macStats_totalPrbUl', 'macStats_macSdusDl_sduLength', 'macStats_macSdusDl_lcid', 'macStats_prbUl', 'macStats_totalPduUl', 'macStats_mcs1Dl', 'macStats_mcs2Dl', 'macStats_prbDl', 'macStats_totalPrbDl', 'macStats_prbRetxDl', 'macStats_totalTbsDl', 'ulCqiReport_sfnSn', 'pdcpStats_pktRx', 'pdcpStats_pktRxW', 'pdcpStats_pktRxAiatW', 'pdcpStats_pktRxOo', 'pdcpStats_pktRxBytesW', 'pdcpStats_pktRxSn', 'pdcpStats_pktTxBytesW', 'pdcpStats_pktTxSn', 'pdcpStats_pktTxBytes', 'pdcpStats_pktRxAiat', 'pdcpStats_pktRxBytes', 'pdcpStats_pktTx', 'pdcpStats_pktTxW', 'pdcpStats_pktTxAiatW', 'pdcpStats_sfn', 'pdcpStats_pktTxAiat', 'rnti', 'quality']*

To identify the fields with the highest correlation with wbCQI, a new list of parameters is created indicating the correlation levels (from 0 to 1) with other fields. From this list (see below), it can be observed that

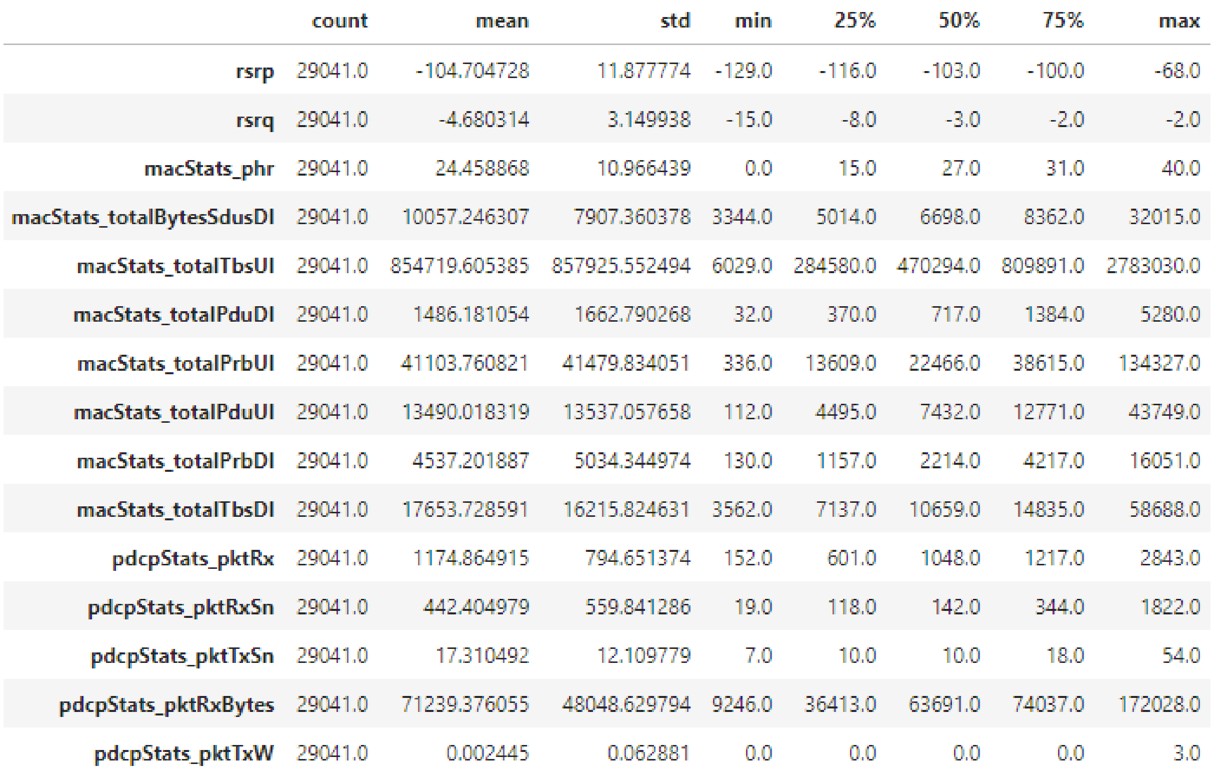
- RSRP and RSRQ have a strong correlation with wbCQI
- macStats_mcs1Dl abd macStats_mcs1Ul correlate quite nicely with wbCQI (which might change on other patterns, in particular for RAN slicing)
- pktRx or pktTx has no correlation at all as they are not related with wbCQI
- RSRP, RSRQ, and PHR correlate strongly among each other.
- tbsDl correlates with with Tx related fields
- prbUl, rnti, date_index, pdcpStats_sfn and other are unnecessary fields that are not needed to generate the model

It should be noted that the mcs1Dl is created and calculated using wbCQI directly. Therefore, it has almost a one to one correlation with wbCQI. Using this field in the training would cause all models to put their coefficients to this field and provide 100% accuracy. However, this field cannot be used as mcs1Dl is obtained from the wbCQI and not vice-versa. Therefore, this field is removed from the dataset.

3.4.2.2 Final Dataset

After removing the unnecessary columns from the dataset, only 15 fields will be used for the prediction (see Table 33). In addition, we removed the wbCQI as the objective is to predict it and therefore, we do not want it in the predictors for the moment.

Table 33 Final RAN optimisation dataset sample



	count	mean	std	min	25%	50%	75%	max
rsrp	29041.0	-104.704728	11.877774	-129.0	-116.0	-103.0	-100.0	-68.0
rsrq	29041.0	-4.680314	3.149938	-15.0	-8.0	-3.0	-2.0	-2.0
macStats_phr	29041.0	24.458868	10.966439	0.0	15.0	27.0	31.0	40.0
macStats_totalBytesSdusDl	29041.0	10057.246307	7907.360378	3344.0	5014.0	6698.0	8362.0	32015.0
macStats_totalTbsUl	29041.0	854719.605385	857925.552494	6029.0	284580.0	470294.0	809891.0	2783030.0
macStats_totalPduDl	29041.0	1486.181054	1662.790268	32.0	370.0	717.0	1384.0	5280.0
macStats_totalPrbUl	29041.0	41103.760821	41479.834051	336.0	13609.0	22466.0	38615.0	134327.0
macStats_totalPduUl	29041.0	13490.018319	13537.057658	112.0	4495.0	7432.0	12771.0	43749.0
macStats_totalPrbDl	29041.0	4537.201887	5034.344974	130.0	1157.0	2214.0	4217.0	16051.0
macStats_totalTbsDl	29041.0	17653.728591	16215.824631	3562.0	7137.0	10659.0	14835.0	58688.0
pdcpStats_pktRx	29041.0	1174.864915	794.651374	152.0	601.0	1048.0	1217.0	2843.0
pdcpStats_pktRxSn	29041.0	442.404979	559.841286	19.0	118.0	142.0	344.0	1822.0
pdcpStats_pktTxSn	29041.0	17.310492	12.109779	7.0	10.0	10.0	18.0	54.0
pdcpStats_pktRxBytes	29041.0	71239.376055	48048.629794	9246.0	36413.0	63691.0	74037.0	172028.0
pdcpStats_pktTxW	29041.0	0.002445	0.062881	0.0	0.0	0.0	0.0	3.0

We have recorded 30000 scenarios with the top 15 fields that correlate the most with wbCQI in our training set, where all of them can be represented as integers making regression models much more reliable and easier to train. Dataset has minimum correlation of 0.415038 for field pdcpStats_pktTxSn and all upwards. As all fields are represented as integers (see table above), there is no need for transformations or even support vector machines to convert anything.

3.4.3 Regression Models and Validation

We modelled several different regression and tree techniques, which will be constructed, explained and validated in the following sub-sections. We picked the best of all approaches and created an ensemble model, which will depict the final model and the end-result.

In general, regression models aim at modelling a linear relationship between one or several independent variables and a dependent variable (target variable). In the RAN optimisation case, the dependent variable is wbCQI, while the independent variables are all the other variables that are kept in the dataset to predict the wbCQI.

Before we start the regression, knowing the best 15 features we have but not knowing their relation with wbCQI, it is a good practice to include their second and third order statistics, namely x^2, x^3, $\sqrt{x}$ and $x^{1/3}$, to the data frame to reveal polynomial relations that might have among each other. This will increase the number of features we have in the table and will make calculations more complex, yet it might increase our accuracy. Therefore, the total feature count in the end now is *15 + 4x15 = 75* fields.

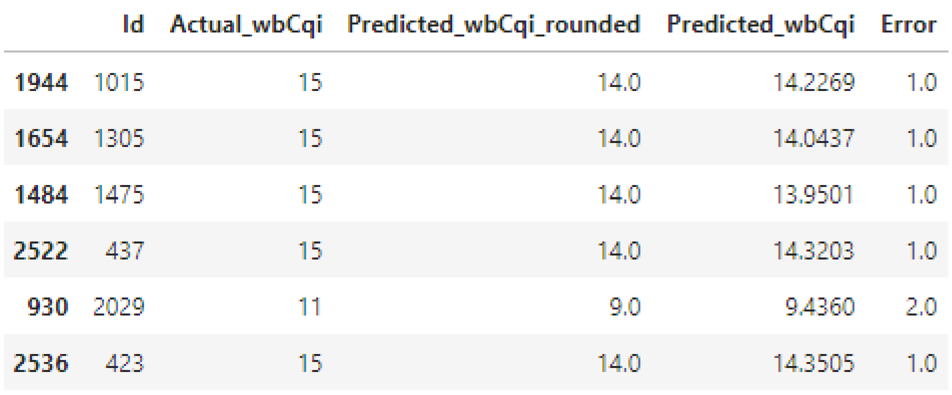
At this stage, we split the dataset to training and validation sets to be sure the validation set is not being used for training purposes. In principle, many techniques involve a percentage-based separator for training and validation sets (i.e 95% to 5% etc). However, we chose to make a scenario based split, where training set includes patterns from different recordings and validation set has its own patterns. This is useful as it will provide more insight of where and when models fail to predict and lose their accuracy. If we would use a percentage-based split, we would not know the exact stats in time series when the casualty started. Still we tried to keep a good ratio of 90% to 10% training to validation set size for this part. In our dataset, the training set consists of 26082 Validation set consists of 2959 measurements.

3.4.3.1 LASSO - Least Absolute Shrinkage and Selection Operator

LASSO [28] is a regression method, which penalizes the absolute size of coefficients, which results in a situation where parameter estimates may be exactly zero. The more penalty applied will shrink the estimates towards zero more rapidly. LASSO is a good approach when dealing with highly correlated predictors where standard regression will usually have very high coefficients because of their high coherence. The LASSO technique is regression analysis method and powerful when we have a large number of features. It is a great model when trying to avoid overfitting and it is also helpful to overcome computational challenges. The Lasso technique works by penalizing the magnitude of coefficients of features along with minimizing the error between predicted and actual observations. These are called regularization techniques.

To apply the LASSO method, few parameters are adjusted such as alpha, tolerance for the optimisation (important as it stops the model), epoch count, and features from our dataset. Sample predictions resulted when applying LASSO method is shown in Table 34. Notice the predictions are rounded up or down to integer values as wbCQI is an integer field.

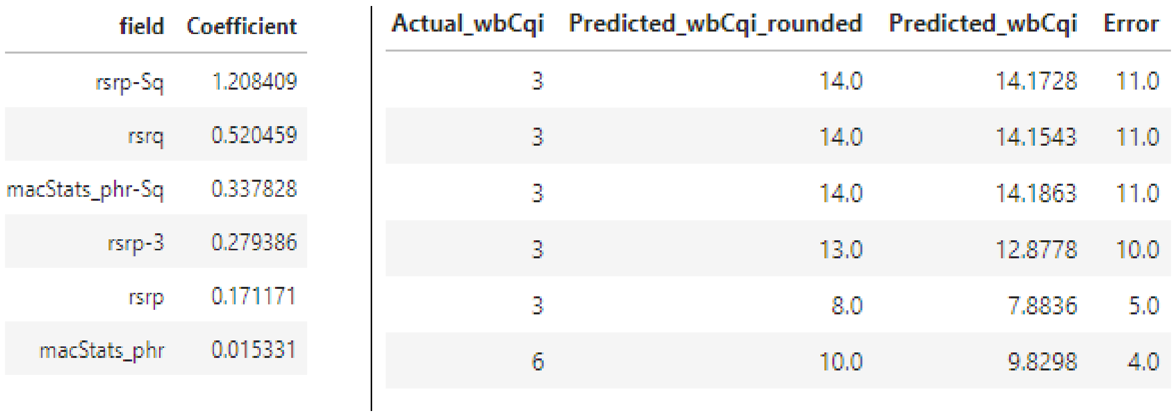
Table 34 Sample wbCQI prediction using LASSO method



	Id	Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
1944	1015	15	14.0	14.2269	1.0
1654	1305	15	14.0	14.0437	1.0
1484	1475	15	14.0	13.9501	1.0
2522	437	15	14.0	14.3203	1.0
930	2029	11	9.0	9.4360	2.0
2536	423	15	14.0	14.3505	1.0

According to the LASSO method, the most important 6 fields for its predictions from our dataset are (see Table 35): Square Root of RSRP, RSRQ, Square Root of PHR, 1/3th power of RSRP, RSRP and PHR. Mean error is 1.239 where around 4 predictions were very wrong, and 689 miss predictions in total of 2959, indicating a %76.71 success rate (see Figure 29).

Table 35 Field coefficient indicator and highest predication error using LASSO method



field	Coefficient
rsrp-Sq	1.208409
rsrq	0.520459
macStats_phr-Sq	0.337828
rsrp-3	0.279386
rsrp	0.171171
macStats_phr	0.015331

Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
3	14.0	14.1728	11.0
3	14.0	14.1543	11.0
3	14.0	14.1863	11.0
3	13.0	12.8778	10.0
3	8.0	7.8836	5.0
6	10.0	9.8298	4.0

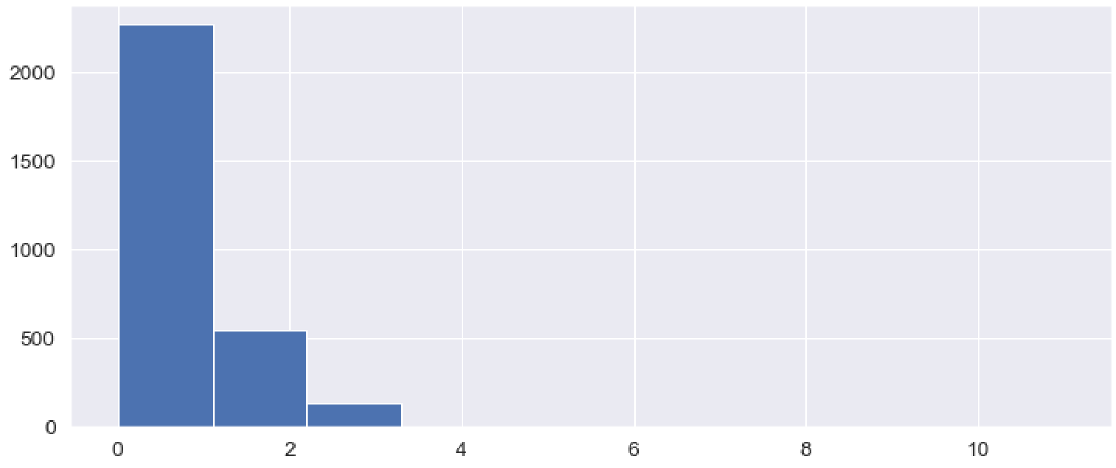


Figure 29 Lasso wbCQI Error

3.4.3.2 Elastic Net

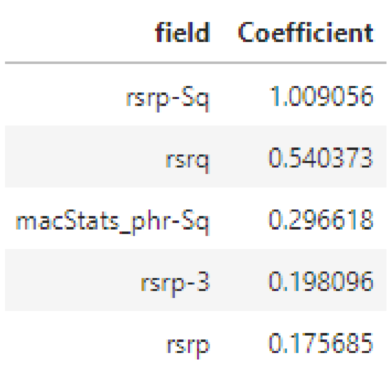
The Elastic Net is a combination of LASSO and ridge regression. It aims at overcoming individual limitations that are prevalent in the LASSO and Ridge, while taking advantage of each model's strengths. The Elastic Net enforces sparsity. Sparsity refers to the relationship between the amount of predictors and the count of samples. If the amount of predictors is greater than or equal to the amount of samples, our model is impossible to fit. Therefore, we use a subset, which is smaller than the number of predictors. This leads to the great advantage, that the Elastic Net does not have a limitation on how many predictors we can use. In our case, with this dataset, the dataset is greater than the number of predictors by far. However, it is nice to take advantage of another strength of the Elastic Net. The Elastic Net encourages a grouping effect amongst highly correlated predictors. This allows for a better control of the impact of each predictor. Table 36 below depicts an example of wbCQI predictions using Elastic Net.

Table 36 Sample wbCQI prediction using Elastic Net method

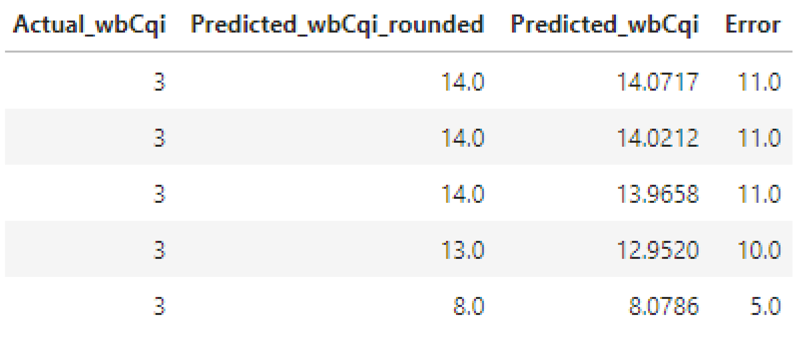
Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
7	10.0	9.7482	3.0
7	10.0	9.6736	3.0
6	10.0	9.6737	4.0
8	10.0	9.6735	2.0
8	10.0	9.6736	2.0

According to Elastic Net model (see Table 37), the most important fields were again square root of RSRP, RSRQ, square root of PHR, 1/3th power of RSRP and RSRQ. Mean error is 1.357 where about 4 predictions were wrong, and 815 miss-prediction with a total of 2959, indicating % 72.45 success rate (see Figure 30).

Table 37 Field coefficient indicator and highest predication error using Elastic Net method



field	Coefficient
rsrp-Sq	1.009056
rsrq	0.540373
macStats_phr-Sq	0.296618
rsrp-3	0.198096
rsrp	0.175685



Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
3	14.0	14.0717	11.0
3	14.0	14.0212	11.0
3	14.0	13.9658	11.0
3	13.0	12.9520	10.0
3	8.0	8.0786	5.0

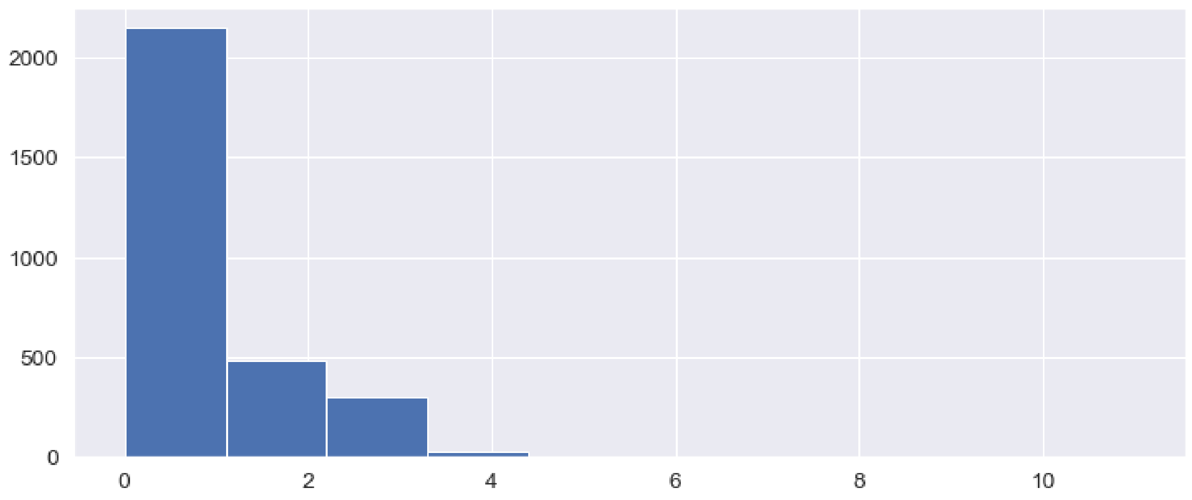


Figure 30 Elastic Net wbCQI Error

3.4.3.3 Tree Based Models - Random Forest and XGBoost

Tree based learning algorithms are one of the best and mostly used supervised learning methods. They have the advantage that they can model non-linear relationships quite well. Random Forest works for both categorical and continuous input and output variables. A tree is said to be categorical if the target variable is categorical. In our case, we are creating a continuous random tree, as the target variable "quality" is continuous. The advantages of random forests are that they are easy to understand, useful in data exploration and not constrained by the data type. Data does not need to be cleaned as intensely for using random forests and random forest is a non-paramedic method, which means that trees have no assumptions about the space distribution and the classifier structure. The disadvantages of trees are that they tend to over-fit and when we use them for continuous variables, the tree loses information when it categorizes variables in different categories. Both disadvantages shall be accounted when modelling wbCQI.

3.4.3.3.1 Random Forest Regressor

Random Forest [29] is a versatile ML method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration, and proved to be a good method. It is a type of ensemble learning method, where a group of weak models combine to form a powerful model.

If we have a number of instances in training set N, each sample these N instances is taken at random but with replacement. This will be the training set for growing our tree. From M input variables, a number m<M is specified such that each node, m variables are selected at random out of M. The best split on m is used to split the node. The value m is then held constant while we grow the forest. Each tree is grown to the largest extent and we do not prune. Lastly we predict new data by aggregating the predictions of the n trees.

In should be noted that the Random Forest implementation actually uses percentage base split of the training set to validate it against itself. The final model than can be used to predict the real validation set to confirm. With Random Forest Regressor, at every run we get a different model with a different accuracy. This has the benefit of stochasticity helping to build dynamic models that are trained on the fly. Figure 31 shows the wbCQI prediction using Random Forest method. It can be observed that the outliers where wbCQI is quite low. Training is randomized and parameters should be optimized better when dataset size increases.

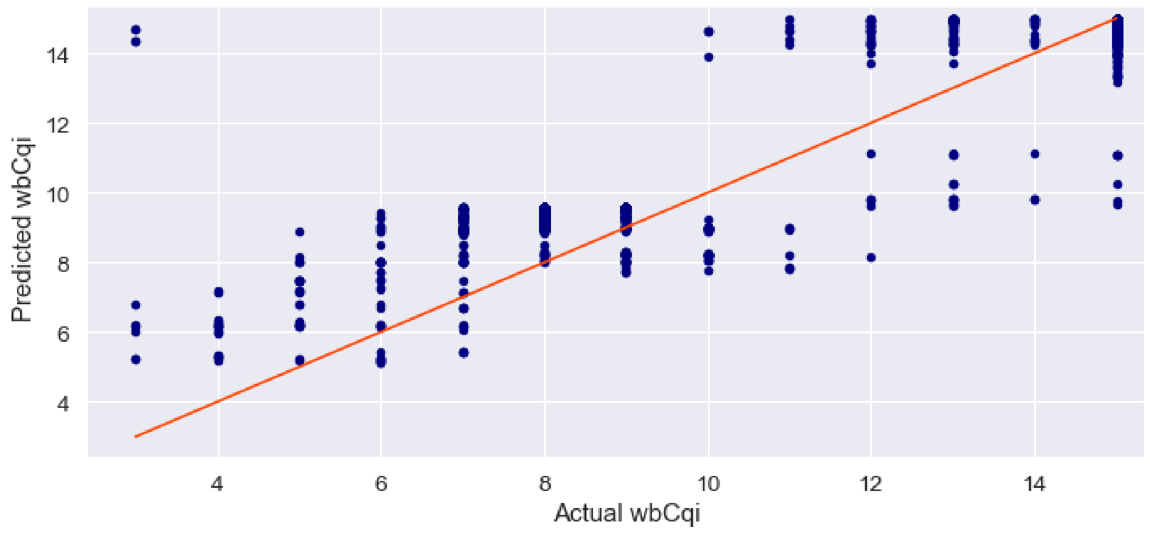
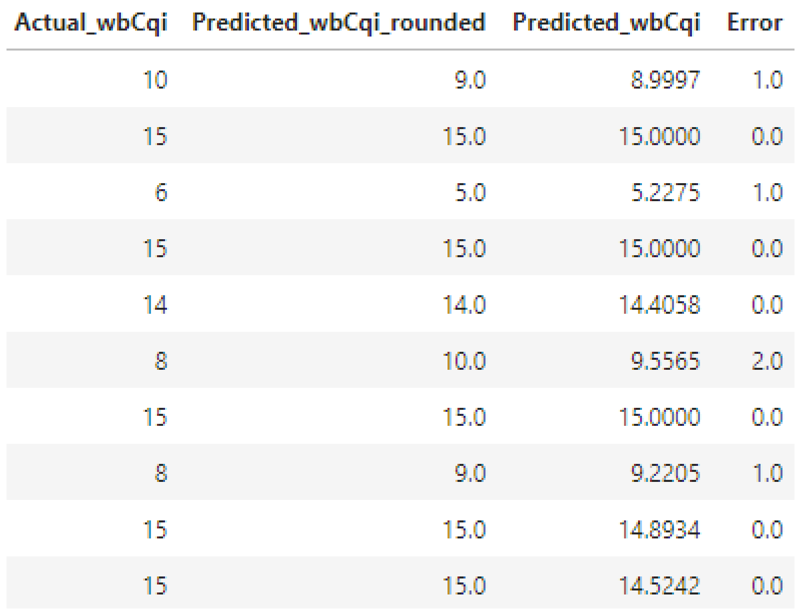


Figure 31 Sample wbCQI prediction using Random Forest method

For a relatively small dataset like the one used here, some sample predictions are shown in Table 38.

Table 38 Sample wbCQI prediction using Random Forest



Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
10	9.0	8.9997	1.0
15	15.0	15.0000	0.0
6	5.0	5.2275	1.0
15	15.0	15.0000	0.0
14	14.0	14.4058	0.0
8	10.0	9.5565	2.0
15	15.0	15.0000	0.0
8	9.0	9.2205	1.0
15	15.0	14.8934	0.0
15	15.0	14.5242	0.0

Mean error is 0.68 which is under 1 and is much better than the previous models with 483 miss predictions in total of 2959, indicating % 83.67 success rate (see Table 39 and Figure 32).

Table 39 Highest predication error using Random Forest method

Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
3	15.0	14.718	12.0
3	15.0	14.592	12.0
3	14.0	14.428	11.0
3	14.0	14.452	11.0
10	15.0	14.534	5.0

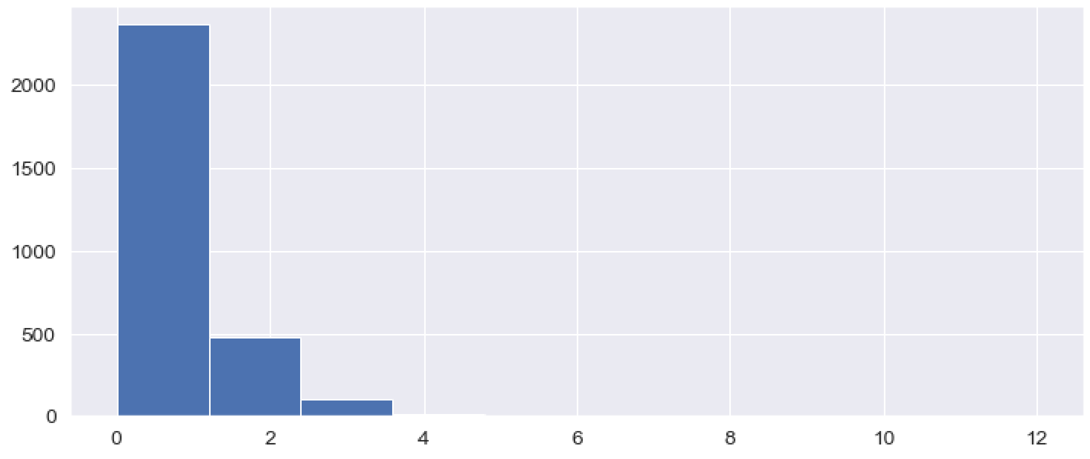


Figure 32 Random Forest wbCQI Error

3.4.3.3.2 XGBoost

XGBoost stands for “Extreme Gradient Boosting” and is another Tree implementation that is used for wbCQI prediction. As in random forest, we grow a tree decision by decision. Yet, the difference of XGBoost lies within Training. As the name of the implementation implies, training is "boosted". We learn each variable-by-variable relation and grow a tree for each. Then we apply something called additive training, where we grow the tree by adding new trees in an iterative way. In our dataset for example, we would grow a tree for RSRP and add it to a tree that we already grew for RSRQ. We need to keep score of each leaf and tree structure. Therefore, we built this model in an iterative way. This is a far more complex optimisation than just optimizing with a gradient. Naturally, we add the trees that optimize our result. Also, XGBoost takes advantage of regularisation methods, which many other implementations do not. It has to be noted that XGBoost has subsampling percentage drop to avoid overfitting. Table 40 below depicts an example of wbCQI predictions using XGBoost.

Table 40 Sample wbCQI prediction using XGBoot method

Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
15	14.0	14.2399	1.0
9	9.0	8.7151	0.0
15	14.0	14.0042	1.0
9	8.0	8.2443	1.0
8	9.0	9.1631	1.0
15	14.0	14.1698	1.0
15	14.0	13.9904	1.0
15	14.0	13.5287	1.0

Mean error is 0.66 (the best results obtained). Only 4 predictions are wrong and model on average performs much better than the others with only 345 miss predictions in total of 2959, indicating % 88.34 success rate (see figure Table 41 and Figure 33).

Table 41 Highest predication error using XGBoost method

Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
3	15.0	14.718	12.0
3	15.0	14.592	12.0
3	14.0	14.428	11.0
3	14.0	14.452	11.0
10	15.0	14.534	5.0

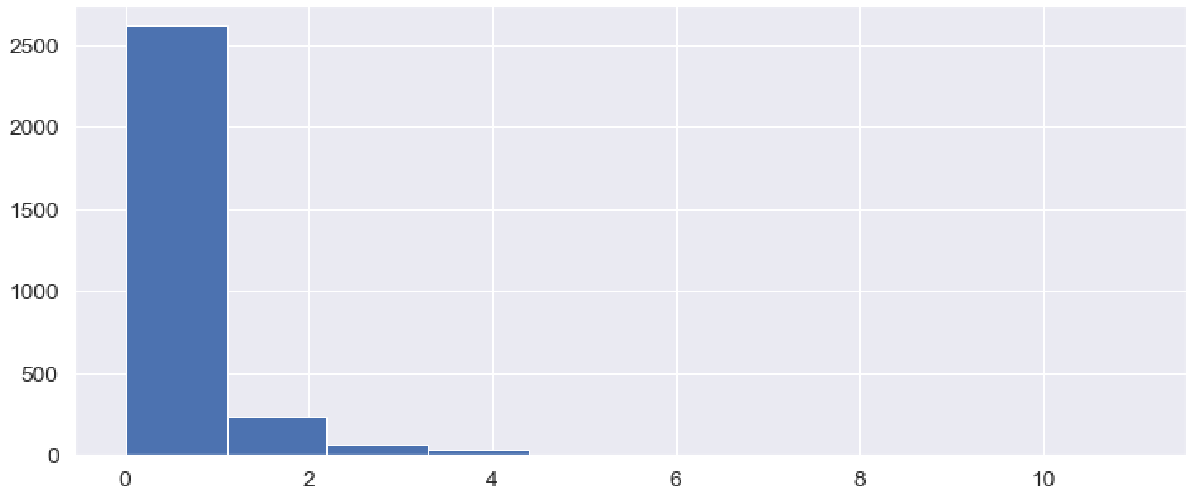
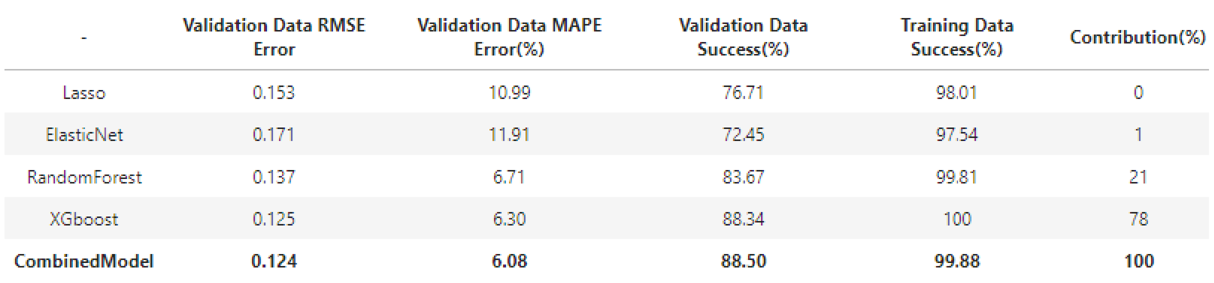


Figure 33 XGBoost wbCQI Error

3.4.3.3.3 Combined Model

Not all the models presented in above sub-sections perform the best under all circumstances. Therefore, we developed an optimizer that will test all contributions from 0% to 100% of all models to combine them together and try to find a contribution level where the validation set will be predicted the best. Combined model concluded that 78% of XGBoost, 21% of Random Forest and only 1% of Elastic Net would provide the best combined model. As expected, XGBoost and Random Forest have the highest contributions as their performances are significantly better. However, the results reveal that regression models shall include methods that achieve lower percentages so as to provide better yet more accurate predictions. Table 42 depicts a sample of the predictions obtained through the combined method.

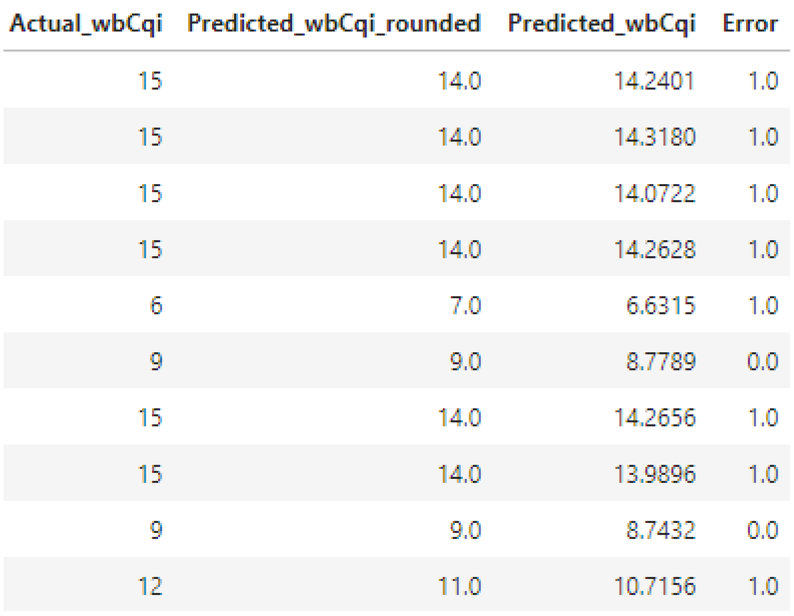
Table 42 Sample wbCQI prediction using Combined method



-	Validation Data RMSE Error	Validation Data MAPE Error(%)	Validation Data Success(%)	Training Data Success(%)	Contribution(%)
Lasso	0.153	10.99	76.71	98.01	0
ElasticNet	0.171	11.91	72.45	97.54	1
RandomForest	0.137	6.71	83.67	99.81	21
XGboost	0.125	6.30	88.34	100	78
CombinedModel	**0.124**	**6.08**	**88.50**	**99.88**	**100**

Mean error is 1.03 which is much higher than expected with 344 miss predictions in total of 2959, indicating % 88.37 success rate (see Table 43). Even though the combined method has the best success rate on the validation data, the mean error remains high, which in turn limits the applicability of model.

Table 43 Highest predication error using Combined method



Actual_wbCqi	Predicted_wbCqi_rounded	Predicted_wbCqi	Error
15	14.0	14.2401	1.0
15	14.0	14.3180	1.0
15	14.0	14.0722	1.0
15	14.0	14.2628	1.0
6	7.0	6.6315	1.0
9	9.0	8.7789	0.0
15	14.0	14.2656	1.0
15	14.0	13.9896	1.0
9	9.0	8.7432	0.0
12	11.0	10.7156	1.0

4 Related 5G Standards

This section reviews the relationship of the SliceNet's Cognition Sub-Plane to related 5G standards.

4.1 Overview

SliceNet Cognition Sub-Plane design and methodology is aligned with several of the main standardization bodies and associated documents in regards to ML and cognition for the management of network systems.

The ETSI Experiential Networked Intelligence (ENI) Industry Specification Group (ISG) focuses on the specification of a cognitive network management system for the optimisation and adjustment of NO systems over the time based on metrics extracted from the underlying network system. Given the current trends on both research and industry, where technologies such as Software Defined Network (SDN), NFV and slicing are of paramount importance, the ENI ISG proposes an architecture that adopts a "observe-orient-decide-act" approach to guide an assisted system and influence in its behaviours by providing recommendations, for example, in the form of policies [30]. Figure 34 (right) depicts a simplified view of the proposed approach. SliceNet Cognition Sub-Plane perfectly matches ENI's architecture; the Cognition Sub-Plane acts as the ENI block, adopting a MAPE-K loop to direct the actions of an assisted system, which is the rest of the SliceNet architecture (see Figure 34 (left)). The communication between the two blocks is also achieved thanks to a collection of APIs for said purpose, allows for the flow of information to the recommendation system with relevant metrics from the assisted system as well as to force the recommendations back to the assisted system. The main difference between the two approaches stems from the way in which the recommendation system influences the assisted system. ENI's approach advocates for a more declarative approach, in which the recommendation block suggests changes on policies or system states, which are then left to the discretion of the assisted system. On the other hand, SliceNet's Cognition Sub-Plane, although is also being governed by a policy system, engages with the assisted system in a more imperative way. More specifically, the QoE Optimizer module, which is responsible for determining the remedial actions once QoE violations occurs or undesired situations are detected, contacts the assisted system (the Orchestration Plane) with the exact action that needs to be carried out, leaving the specific details on how to achieve this to the logic of the assisted system. This difference, although subtle, allows SliceNet to be more focused on its goals, which from the perspective of the Cognition Sub-Plane, is the QoE-aware management of NSes as offered to vertical clients.

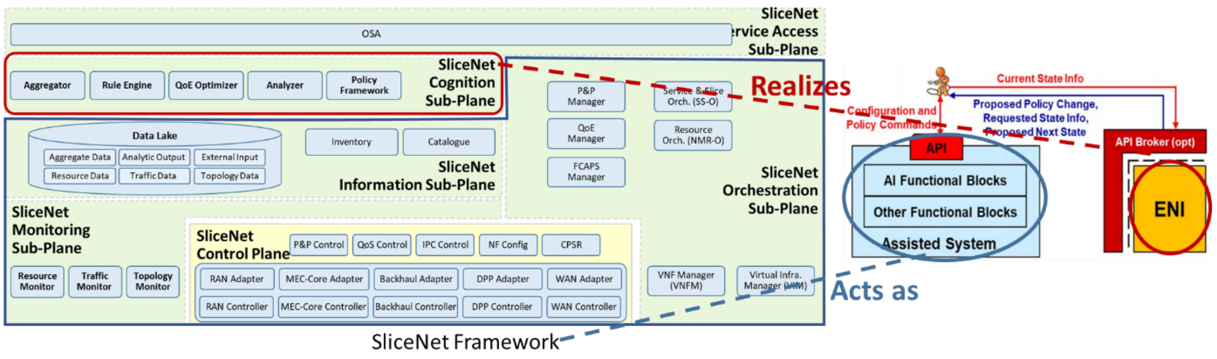


Figure 34 Mapping between SliceNet's Cognition Sub-Plane and ETSI ENI architecture

The NGMN also defines as one of the key requirements for 5G systems the concept of Autonomic Networking (AuN). Given the complexity of 5G systems, in which multiple network technologies and service requirements coexist, it is essential to push the concept of AuN from an E2E perspective for automating the configuration, management, operation and awareness of 5G systems. For this, in [31], NGMN introduces the presence of Autonomic Management and Control (AMC) in the overall

E2E 5G network architecture. Similar to ENI, it proposes a framework in which an underlying network infrastructure layer is governed by another entity that provides insights about the overall system and then enforces actions to the infrastructure layer. In this regard, NGMN divides the said entity into two different roles (Figure 35 (right)). First, an Automated Management layer, in which reasoning, learning and adaption procedures take place. Then, the created insights are distributed to an AMC layer, which focuses on the efficient implementation and automation of configuration and monitoring workflows. The key elements of the AMC layer are the network Decision Entities (DEs), which, given the inputs disseminated from the Automated Management layer (e.g. policies), autonomously take decisions for self-configuration, self-healing, self-diagnosis, etc. of the network layer. Lastly, a common Knowledge Base (KB) is present for the sharing of data between the layers. SliceNet's Cognition Sub-Plane combines both layers capabilities, in which insights are gained through specialized analytical functions, overall behaviours are defined thanks to policies and (re-)configurations are determined through the capabilities of a dedicated DE, namely the QoE Optimizer. The difference between the two approaches, aside from the separation/aggregation of roles, resides in the specialization of SliceNet's Cognition Sub-Plane, which focuses on the management of NSes rather than general network management. Nevertheless, the two approaches are perfectly compatible, highlighting the alignment of SliceNet with 5G architecture standards.

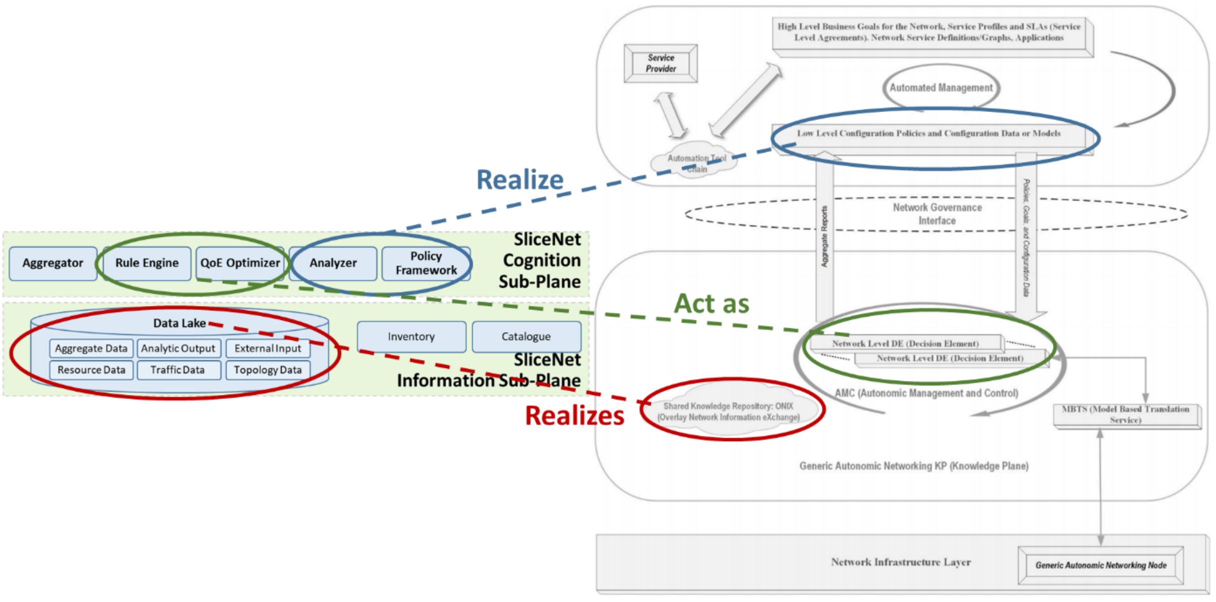

Aggregator
Rule Engine
QoE Optimizer
Analyzer
Policy Framework
SliceNet Cognition Sub-Plane
Data Lake
Aggregate Data
Analytic Output
External Input
Resource Data
Traffic Data
Topology Data
Inventory
Catalogue
SliceNet Information Sub-Plane
Realize
Act as
Realizes
Service Provider
High Level Business Goals for the Network, Service Profiles and SLAs (Service Level Agreements). Network Service Definitions/Graphs, Applications
Automated Management
Low Level Configuration Policies and Configuration Data or Models
Automation Tool Chain
Network Governance Interface
Aggregate Reports
Policies, Goals and Configuration Data
Network Level DE (Decision Element)
Network Level DE (Decision Element)
AMC (Autonomic Management and Control)
Shared Knowledge Repository: ONIX (Overlay Network Information eXchange)
MBTS (Model Based Translation Service)
Generic Autonomic Networking KP (Knowledge Plane)
Network Infrastructure Layer
Generic Autonomic Networking Node

Figure 35 Mapping between SliceNet's Cognition Sub-Plane and NGMN AMC framework

In regards to data-driven network operations, control and management, ITU-T specifies a generic framework for big-data-driven networking (bDDN) in [32]. In this document, the ITU-T defines a three-plane framework. Rather than a traditional layered framework, the proposed framework and its planes engage in a tri-dimensional manner for network operation and management, in which the three defined planes – big-data, management and network planes (see Figure 36) – take joint actions and responsibilities for the management of a network infrastructure in the presence of enormous sources of data which may provide valuable insights to govern the network operation. The main protagonist of the bDDN framework is the big data plane. The inclusion of such plane arises from the fact that traditional control and management layers do not properly handle the challenges of big data, such as data aggregation, analysis, federation and storage. With this plane as the central element, different bDDN domains are defined, representing a network system in which management and control operations are governed thanks to big-data-based analysis and reasoning. SliceNet's Cognition Sub-Plane follows such an approach, with the developed Data Lake acting as the big-data plane. Then, the responsibilities of big-data-based analysis and operation are distributed across the multiple elements of the Cognition Sub-Plane. In this regard, SliceNet combines some of the capabilities of the big-data plane and the management plane into a single plane, namely the Cognition Sub-Plane. This arises from the fact that some of the operations related to bDDN do not have a clear separation of roles as all the involved elements need to benefit from the shared knowledge and analysis. Thus, SliceNet combines both big-data monitoring, analysis and operation-based into the same plane to achieve a more holistic data-driven management of slices. In addition, it further evolves the standard by not focusing on generic networking but rather embracing network slicing and providing solutions for the specific challenges that such technology implies.

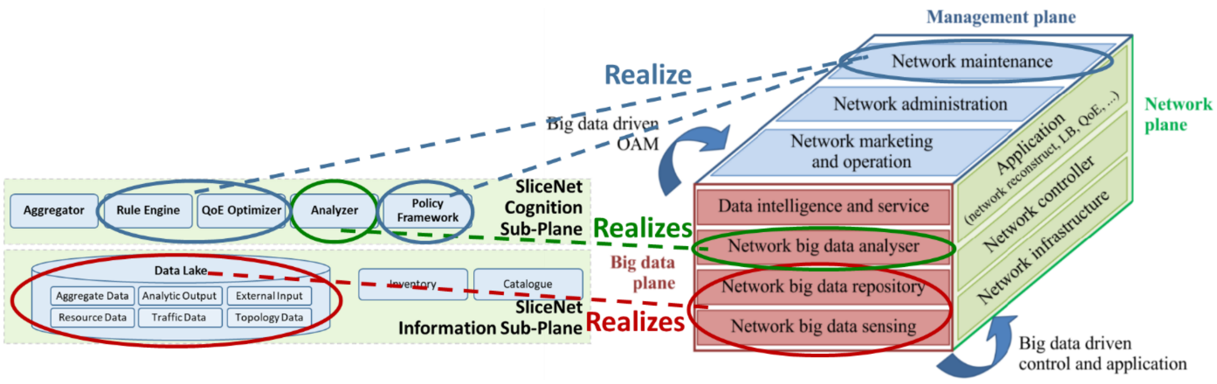


Figure 36 Mapping between SliceNet's Cognition Sub-Plane and ITU-T bDDN framework

4.2 In-Progress Contributions to Standards

4.2.1 ITU-T Alias

Two contributions of the WP5 have been submitted to the standards organization ITU-T Focus Group ML5G under study group 13 related to two ML models developed within SliceNet project. The submitted contributions are the following:

1. Anomaly prediction and integration for eHealth UC based on vertical feedback.
2. Noisy neighbour detection and integration in a virtualized infrastructure.

During the ITU FG ML5G web-conference, held the 20th November 2019, we presented the two proposed contributions and an overview of SliceNet architecture and of the cognitive closed loop. Additionally, the group expressed their interest in the inventories and about the control plane of SliceNet, the interfaces, the APIs, and the vertical feedback. The next step will be to update the contributions in order to express more in details the following topics:

- Defining the vertical input and feedback.
- Mapping the vertical feedback to Slice parameters.
- Detailing the APIs for data monitoring and actuation via SliceNet control plane.

4.2.2 ETSI INT AFI

WP5 submitted a document (contribution number INT(19)044010) about SliceNet architecture with focus on the Cognition Sub-Plane to the standards organization ETSI TC INT AFI (working group on Autonomic Management & Control). The presentation has been done during ETSI INT #44 meeting. The group expressed their interest in the work related to comparison and alignment between Generic Autonomic Network Architecture (GANA) and SliceNet Architecture. If we can agree on the alignment, a new Work Item can be opened at ETSI TC INT to work on a new Technical Report.

5 Related 5G projects

This section reviews the SliceNet's Cognition Sub-Plane compatibility with other 5G projects and its innovations beyond these projects. During Phase I of the 5GPPP programme, several projects addressed the broad concept of Self Organizing Networks (SONs), in which ML/cognition-aided network management is enclosed. In particular, three projects dealt specifically with network management, being Sesame, SelfNet and CogNet.

Starting with Sesame [33], this project has been devoted to the management of multi-tenant small radio cells. It defines a Cloud Enabled Small Cell (CESC) that combines the capabilities of traditional physical SCs with data centre (DC) capabilities to enhance the virtualization of the Physical Network Functions (PNFs) at the network edge. Then, a CESC Manager (CESCM) entity is introduced to allow for the management of the cells and its elements. To enable a flexible management, the CESCM integrates several SON capabilities, mainly SLA monitoring, to stimulate the multiple Element Management Systems (EMSs), which will operate autonomously the underlying network functions with the goal of SLA maintenance. Although Sesame follows the SON approach for self-operation of the network, it does not employ ML/cognition techniques to gain insights on the monitored SLAs and help to make smart decisions. SliceNet does follow the SON principles, in which several elements automate the operation of the underlying physical/virtual network. In addition, SliceNet incorporates several ML/cognition-based elements, i.e. the Cognition Sub-Plane, to enhance the SON capabilities, gaining better insights about the network state and providing proper context for the automated decisions.

Both projects SelfNet [34] and CogNet [35] incorporate ML capabilities to enhance the management of the underlying infrastructure. In particular, SelfNet is devoted to the management of key network technologies for 5G systems, including SDN and NFV-enabled networks, with a special focus on SONs for self-monitoring, self-optimisation, self-protection and self-healing. Both its control and orchestration components are enriched with some analysis capabilities (based on ML) that allow for a better self-control/management of the network and data layers. Nevertheless, SelfNet is not fully dedicated to the cognitive management of networks. In this regard, CogNet is totally dedicated to the cognitive management of network infrastructures, with a special emphasis on the NFV aspect of the underlying substrate. To this end, CogNet defines a complete ML-based framework that provides monitoring, analysis and actuation capabilities to influence on the operation of an NFVI. Similar to the methodology followed in SliceNet, CogNet also follows a data-driven and policy-based framework for ML/cognition. The main difference between the two is that CogNet is focused on the generic cognitive management of an NFVI while SliceNet's Cognition Sub-Plane is focused on the QoE-aware management of 5G networks, which requires specialized analysis and actuation components to provide the necessary context. Although both projects expand the SON framework with cognition capabilities, their scope is significantly different. SelfNet focuses on the networking aspects of a 5G system while CogNet focuses on the NFV side of the infrastructure. Beyond these aspects, SliceNet is devoted to the E2E cognitive management of NSes build on top of a 5G system, including both networking and NFV aspects. Not only that, SliceNet Cognition Sub-Plane takes special care to incorporate the slice/vertical context in its analysis and actuations so as to fully provide a QoE-aware management of NSes.

In summary, SliceNet Cognition Sub-Plane has further expanded the architectural proposals of past projects by going beyond a simple network infrastructure but encompassing the E2E aspect of the service delivery, by tackling the cognitive management of NSes, an aspect that has not been tackled in Phase I. Moreover, a key aspect of the developed Cognition Sub-Plane is the "verticals-in-the-loop" philosophy adopted by SliceNet, which makes the verticals an integral part of the full system. Due to that, the multiple analysis functions of SliceNet's Cognition Sub-Plane are specialized for the QoE monitoring/analysis of verticals' slices, with properly curated monitored data to reflect the requirements of the managed slices. Figure 37 depicts a schematic of the position of the different projects that summarizes their respective scope related to cognitive management, SON, and their infrastructure focus.

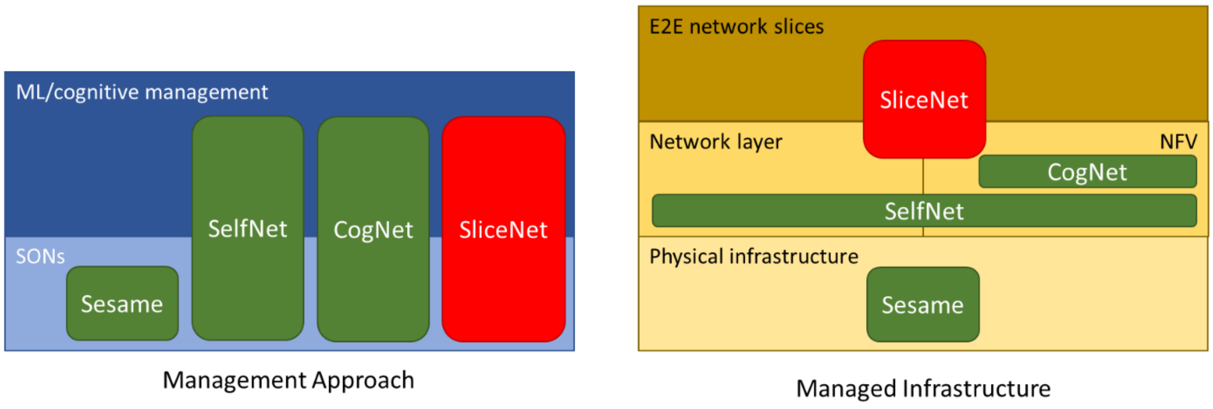


Figure 37 Comparison of management and managed infrastructure scopes

6 Cognition Sub-Plane and the SliceNet Architecture

As shown in previous deliverables D5.5 [1] and D5.6 [2], as well as in Section 2 of this deliverable, SliceNet Cognition Sub-Plane has accomplished its goal to provide a framework for the QoE-aware management of NSes on top of a shared 5G network infrastructure. Two main developments have been key for achieving this objective: the development of ML analytical models to gain insights about the QoE of deployed slices and the development of an actuation framework devoted to the maintenance of quality levels per slice.

Multiple ML models serve as advanced QoE sensors that determine relevant metrics and KPIs to determine the quality of the underlying slices or events/changes on the infrastructure that may affect the health of the slice. In this regard, SliceNet Cognition Sub-Plane has contributed to the progress of QoE awareness/analysis of vertical slices by analysing the requirements of selected vertical UCs and mapping its requirements onto metrics at the slice level that may affect their quality. This work can help with other UCs related to similar vertical sectors as samples of the metrics to be employed for QoE monitoring of slices supporting specific vertical services. In addition, aside from the specific models developed for SliceNet's vertical UCs, SliceNet's Cognition Sub-Plane also focused on the generic aspect of QoS to QoE classification, which is a very relevant to optimizing QoE-aware management of NSes. Indeed, current Internet service trends (e.g. Youtube, Skype) usually employ a QoE rating to classify the quality of the service. Then, it is up to the service/network operator to determine, in case of bad quality, which network parameters are responsible for the degraded service (i.e. QoS metrics). Such a task is usually complex, as there is not a direct correlation between user QoE and network QoS metrics. Hence, the model developed within SliceNet's Cognition Sub-Plane for this objective (the QoS to QoE classification model) sets a solid framework and methodology that could be expanded and contextualized to tackle the task of correlating QoE and QoS in other vertical use cases.

On the other side of the equation, lays the actuation framework of SliceNet's Cognition Sub-Plane. Thanks to the two main components – the PF and the QoE Optimizer – SliceNet's Cognition Sub-Plane is able to detect events that effect the QoE of slices (mainly through the outputs of the multiple analytical functions), determine if certain conditions are met and apply remedial actions to overcome undesired situations. This is possible thanks to the policy system, which provides a great deal of flexibility on how the management of QoE per slice happens. Following an ECA model, it allows for mixing and matching monitoring/analysis outputs to specific remedial actions for a plethora of cases. In this regard, SliceNet's Cognition Sub-Plane could be easily expanded to support other cases of QoE-aware management by properly setting new policies that would automate the whole actuation workflow.

These two big elements (analysis and actuation) can be easily combined thanks to the Data Lake approach followed by SliceNet's Cognition Sub-Plane. Indeed, the Data Lake approach has been a key design feature for the successful development and prototyping of the Cognition Sub-Plane. By sharing and aggregating the monitoring data coming from the NSP level, it is possible to centralize the queries of analytical functions (i.e. ML models), which then feedback their outputs to the centralized Data Lake for the consumption by the QoE Optimizer. Such an approach not only allows for a relative agnostic analysis-to-actuation workflow, in which analytical functions only have to be concerned with inserting elaborated data back to the Data Lake and the QoE Optimizer only has to be concerned with extracting said data from the Data Lake, but it also allows for expanding the types of analysis and actuation that can happen in the SliceNet's Cognition Sub-Plane. Indeed, new analytical functions can be easily incorporated into the picture by properly configuring which data would be consumed from the Data Lake and how the generated outputs are inserted back as elaborated events. Then, thanks to new policies, the QoE Optimizer would be made aware of the new data that it needs to query from the Data Lake and the expected action in cases where adverse conditions are inferred from the new inputs feed into previously generated ML models. Figure 38 depicts an example of this environment, showcasing how previous deployed analytical functions and actuations can coexist with the inclusion of new ones. This approach can be exploited by future developments in order to tackle the full loop of QoE-aware management related to other vertical UCs that were not originally under the scope of the SliceNet project.

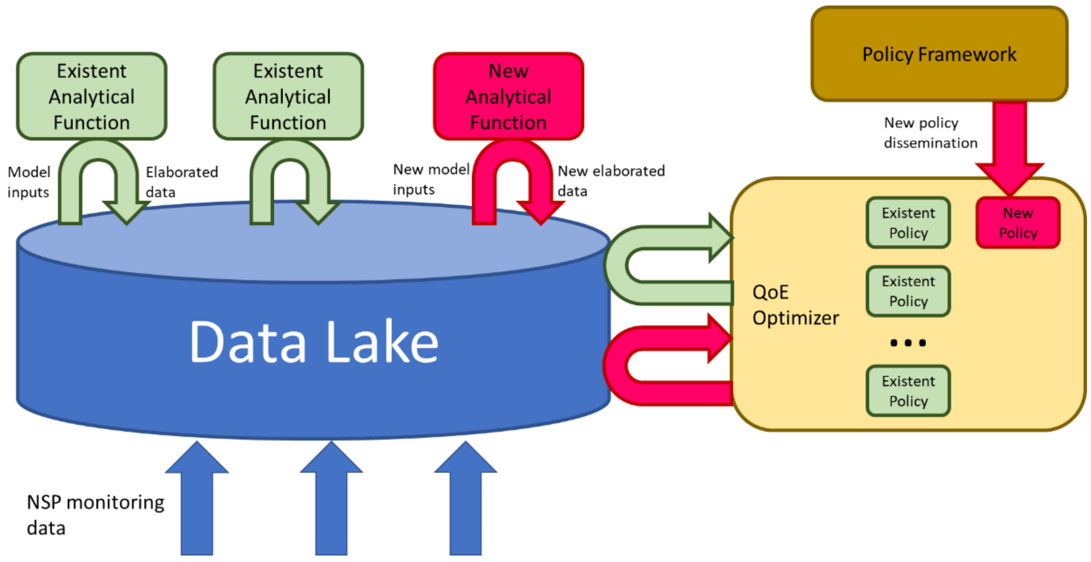


Figure 38 Inclusion of new QoE-aware management strategies thanks to the Data Lake approach

To further enrich the capacities of the developed Data Lake approach, SliceNet's Cognition Sub-Plane has developed, in collaboration with the Plug and Play (P&P) controller developed within WP4 [36], a vertical's feedback mechanism that allows for the vertical customers of deployed slices to express their experience with the provisioned infrastructure. This mechanism has been one of the main innovations developed within the SliceNet project to progress towards the inclusion of the verticals' in the whole management/control of deployed slices. Figure 39 depicts a schematic of the whole feedback mechanism, including the primary components and their interactions.

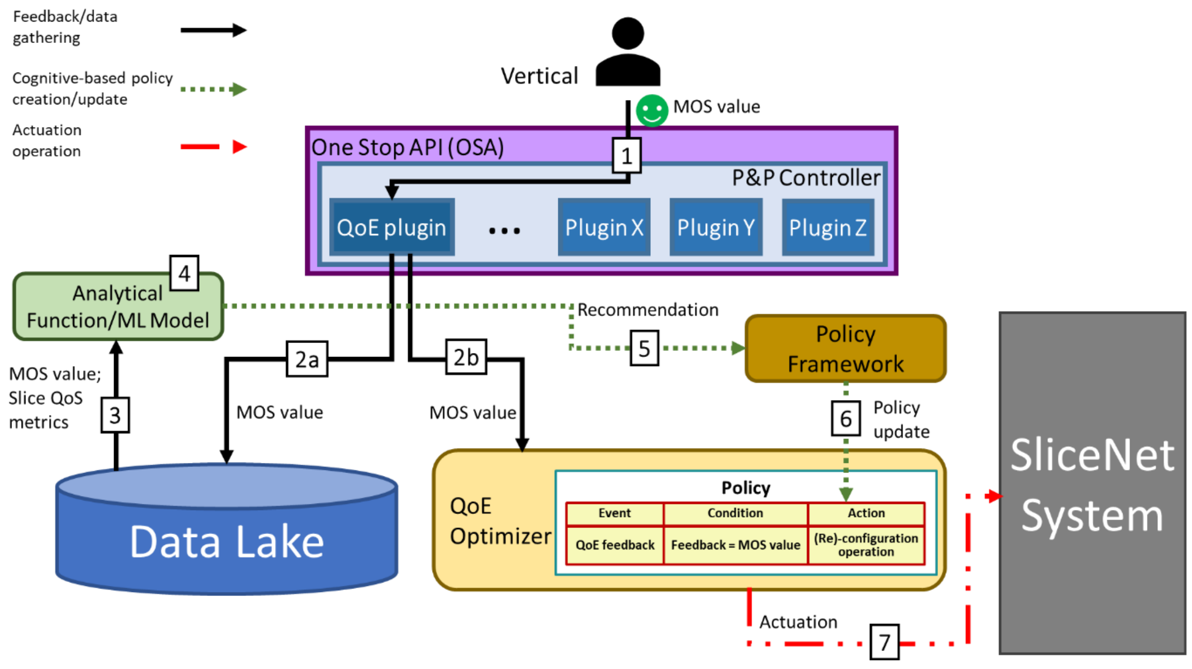


Figure 39 Vertical feedback mechanism workflow

Thanks to the capabilities exposed through the OSA, which grants a vertical customer access to the SliceNet's ecosystem, this feedback mechanism allows collecting a qualitative or quantitative rating – a Mean Opinion Score (MOS) value – from the vertical through a dedicating function enclosed within a specialized plugin (QoE plugin) in the P&P controller of the slice (Step 1). The collected rating can then be captured by the corresponding instance of the QoE Optimizer (Step 2b) which, as explained before in previous WP5 deliverables, has several policies in place that dictate under which **E**vent and **C**ondition what **A**ction should be carried out to remedy bad quality situations. In the case of a vertical feedback, the policy would state that, given a specific MOS value (e.g. "Bad"), the QoE Optimizer should trigger the specified (re-)configuration operation towards the rest of the SliceNet System (i.e. the DSP Orchestration system) so as to maintain optimal quality levels (Step 7). This policy is initially distributed from the PF upon instantiation of the slice and all software artefacts responsible for its control and management. The parameters of the policy (i.e. the ECA items) would be configured manually through the Policy Administration Point (PAP) according to the requirements of the slice and the service it needs to support.

Indeed, by setting up policies which tie the vertical's feedback with an action in turn (re-)configures the most relevant QoS parameters that are affecting the expressed QoE, it is possible to fully realize the "verticals-on-the-loop" vision that SliceNet is advocating. In this regard, SliceNet's Cognition Sub-Plane has proposed a novel framework to incorporate the presence of vertical customers into the QoE management of slices, one of the key requirements for 5G infrastructures. Due to the attractive nature of the mechanism, SliceNet is currently engaging with standardization bodies so as to promote the feedback framework as part of future standards, as explained in Section 4.2. To fully realize the concept, it would however require the proper analysis of the received QoE feedback in order to influence the policy system and the related actions to apply the most suitable (re-)configuration according to current slice status. Such a workflow is also represented in Figure 39. Starting from the QoE plugin, the collected MOS value would need to be inserted onto the shared Data Lake (Step 2a). Upon insertion, a specialized analytical function/ML model would collect the value along with the QoS metrics of the affected E2E slice (Step 3). The goal of such a model is to analyse the expressed feedback and the monitored QoS at the slice level, in order to identify the main cause(s) for unsatisfactory quality levels (Step 4). After this analysis, the model would then recommend to the PF which QoS parameter(s) need to be influenced as part of policies actions (Step 5). Given these gained insights, the PF would then update the Action field of the policies related to the expressed vertical feedback for the PDPs responsible for slice quality maintenance, i.e. the QoE Optimizer (Step 6). With this updated policy, the QoE Optimizer would then be able to trigger the most suitable action according to the most updated insights gained through the expressed feedback from the vertical once received, following the same procedure as in Step 7. The current implementation of the Cognition Sub-Plane does not account for this dynamic update of policies given analysis of verticals' feedback. Nevertheless, SliceNet's Cognition Sub-Plane has proposed a novel framework to incorporate the presence of vertical customers into the QoE management of slices, one of the key requirements for 5G infrastructures. Hence, future developments and research projects could adopt the developed strategy to further integrate verticals into the full provisioning and management loop of 5G networks.

The aforementioned feedback mechanism is a concrete example of how SliceNet's Cognition Sub-Plane architecture and design supports the insertion of external data to the SliceNet ecosystem. The presence of data is essential for the cognitive management of a network system; we learn from big data research that when more data is made available that ML can be leveraged to derive better and more accurate insights. Depending on the size and types of management that are pursued, it may happen that the network system under direct control of the provider does not generate enough data, derived from their monitoring functions, to perform ML-based operations and gain insights about the system behaviour and operation. As such, it becomes essential to introduce other sources of data to complement the data captured within the system. In this regard, SliceNet approach has been to open up its data warehouse (i.e the Data Lake) to the inclusion of other sources of data; the data sources could be from the users of the deployed services (verticals) or other network systems (as the case explained in Section 2.2). The data management in the developed Cognition Sub-Plane is focused on gaining insights per deployed NS, as such, the multiple sources of data allow for providing a more accurate context for the per NS management. Indeed, this last aspect, the per NS management, has been the main focus of the developed Cognition Sub-Plane. The designed and developed architecture has been structured in a way in which all of its elements – analysis, aggregation, policies and actuations – can be combined in a versatile way which allows for the management of completely independent NSes, capturing slice subjective attributes such as QoE levels, which are slice/service dependant. In a nutshell, SliceNet's Cognition Sub-Plane has devoted its efforts in developing a framework that enables 5G infrastructure systems with the capacity of cognitive management of NSes, with a special focus on the QoE aspects. The flexible engagement of all the elements within the Cognition Sub-Plane allows for tackling multiple vertical UC, as it has been demonstrated in past WP5 deliverables and reiterated in Section 2 of this document.

Last, but not least, SliceNet's Cognition Sub-Plane has developed a flexible framework for QoE-aware management of NSes that may suit the requirements of multiple roles and administrative entities. As it has been demonstrated along the vertical UC presented in Section 2, the components of the Cognition Sub-Plane may be instantiated at both NSP and DSP levels independently, potentially articulating isolated MAPE-K loops per administrative role (e.g. Smart Grid UC), a hybrid MAPE-K loop spanning all the roles (e.g. Smart City UC) or only focusing the cognitive management in one of the roles (eHealth UC). This is possible thanks to the modularity and replicability of the designed Cognition Sub-Plane, that allows to instantiate selected components within the plane per administrative role, all following the same architectural principles that tie their functionalities together. In addition, the followed Data Lake and policy-based design facilitates the ease of adoption of selected components depending on the roles and their interests. From a functional and architectural perspective, the Cognition Sub-Plane and their components are the same in all the roles. Then, the specific behaviour and goals become defined through the data exchanges flows between modules and the policies that govern the Actuation Framework. Such flexible design can be followed in future developments and research projects as a framework capable to adapt to a plethora of vertical UC in which several combinations of administrative roles may take place.

7 Conclusion

The development and prototyping of SliceNet's Cognition Sub-Plane have been a challenging task that required a meticulous analysis of the required functionalities and posterior refinement of the designed architecture. One of the main challenges during its development was the definition of the overall Cognition Sub-Plane architecture since it needed to account for all the requirements posed by the vertical UCs that are supported. In addition, due to refinements of the overall SliceNet architecture and role separations, the Cognition Sub-Plane needed to be adapted in order to match its expected functionalities with the rest of the building blocks and planes of the general architecture.

Speaking about the proposed design, the Data Lake approach has been fundamental in the success of WP5. The separation between functions allowed different partners within WP5 to focus their efforts in concrete parts of the Cognition Sub-Plane without overlapping each other. Once modules where fully developed, the integration consisted of agreeing on how the data flows are realized between analysis and actuations by means of the shared Data Lake and what actuations are needed by means of specifying proper policies. The development of analytical functions also required a strong knowledge of the requirements of vertical UCs, since QoE management of the slices must take the verticals' context in mind. For this, the execution of specific task forces within WP5 to address the challenges of the analysis needed, as well as the actuations, policies and necessary integrations, per TUC enclosed within vertical UC has been useful.

In regards of future work, two main open issues remain when facing the designed and prototyped Cognition Sub-Plane. On one hand, in order to further automatize and make more intelligent the actuations for QoE maintenance, a dynamic and autonomous policy system is required. This can be possible with the engagement of the PF with specialized analytical functions which could recommend policies to be issued towards the QoE Optimizer or even specify new ones by parametrizing certain fields of the policies thanks to the gained insights. The cognitive management/creation of policies is a very challenging task, which needs a careful definition of the policies semantics and parameters and how these can be influenced or modified by ML functions. Nevertheless, despite its challenges, it is an interesting approach to be followed in the future.

The other item that remains open is the specification of a common information model that allows representing analytical functions and available actuations. Such information model would help in further automating QoE monitoring-to-actuation process by enhancing the interaction between ML models and the QoE optimizer, allowing for a dynamic tuning of the desired actuations. In addition, by standardizing the representation of analytical and actuation functions it would be possible to easily incorporate new analysis and actuations on the overall Cognition Sub-Plane framework, fully realizing the benefits of the Data Lake approach which, as explained before, allows for a lose coupling of analytics and actuations, make it possible to grow in capabilities as needed. For these reasons, future works are encouraged to follow this research and development line.

References

[1] SliceNet Deliverable D5.5, "Modelling, Design and Implementation of QoE Monitoring, Analytics and Vertical-Informed QoE Actuators- Iteration I", SliceNet Consortium, March 2019.

[2] SliceNet Deliverable D5.6, "Modelling, Design and Implementation of QoE Monitoring, Analytics and Vertical-Informed QoE Actuators- Iteration 2", SliceNet Consortium, September 2019.

[3] SliceNet Deliverable D7.2, "Prototyping of Vertical Business Use-Cases", SliceNet Consortium, December 2019.

[4] Andersen, B., and T. Fagerhaug, "Root Cause Analysis: Simplified tools and techniques". Milwaukee: ASQ Quality Press, 2000.

[5] C. Cortes and V. Vapnik, "Support-vector networks", Machine Learning, vol. 20, no. 3, pp. 273–297, September 1995.

[6] F. T. Liu, K. M. Ting, and Z. hua Zhou, "Isolation forest," in In ICDM 08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. IEEE Computer Society, pp. 413–422.

[7] WordPress, https://wordpress.com

[8] Kubernetes, https://kubernetes.io

[9] Skydive Open Source Project, http://skydive.network

[10] iPerf3, https://github.com/esnet/iperf

[11] Jupyter Notebooks, https://jupyter.org

[12] Load Testing With Jmeter on Kubernetes and OpenShift tool, https://blog.kubernauts.io/load-testing-as-a-service-with-jmeter-on-kubernetes-fc5288bb0c8b

[13] IBM Watson Studio, https://cloud.ibm.com/catalog/services/watson-studio

[14] Python Data Analysis Library (Pandas), https://pandas.pydata.org

[15] Python Scikit-learn, https://scikit-learn.org/stable

[16] Scikit MLPClassifier, https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[17] TPOT, https://epistasislab.github.io/tpot/

[18] Scikit Logistic Regression, https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[19] Scikit DecisionTreeClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[20] Scikit KNeighborsClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[21] Scikit LinearDiscriminantAnalysis, https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

[22] Scikit RandomForestClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[23] Scikit GaussianNB, https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[24] Scikit SVC, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[25] Scikit AdaBoostClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[26] Scikit QuadraticDiscriminantAnalysis, https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html

[27] Crawdad, https://crawdad.org/eurecom/elasticmon5G2019

[28] LASSO, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoCV.html

[29] Random Forest Regressor, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[30] ETSI, “Experiential Networked Intelligence (ENI); System Architecture”, GS ENI 005 v1.1.1, September 2019.
[31] NGMN, “5G End-to-End Architecture Framework v3.0.8”, August 2019.
[32] ITU-T, “Framework of big-data-driven networking”, ITU-T Recommendation Y.3650, January 2018.
[33] “SESAME: Small cEllS coordinAtion for Multi-tenancy and Edge services”, http://www.sesame-h2020-5g-ppp.eu/
[34] “SelfNet: Framework for Self-Organized Network Management in Virtualized and Software Defined Networks”, https://selfnet-5g.eu/
[35] “CogNet: Building an Intelligent System of Insights and Action for 5G Network Management”, http://www.cognet.5g-ppp.eu/
[36] SliceNet Deliverable D4.1, “Plug & Play Control Plane for Sliced Networks”, SliceNet Consortium, October 2018.