UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dept. of Communications

Analysis of AI/ML algorithms for the management of open 5G mobile networks: xApps in O-RAN

Master's Thesis

Master of Science in Telecommunication Technologies, Systems and Networks

AUTHOR: Boukhayar Bouazza, Dounia

Tutor: Gómez Barquero, David

Experimental director: MOLNER SIURANA, NURIA

ACADEMIC YEAR: 2021/2022

Objectives – The main objective of this work is to analyze AI/ML-based applications for the intelligent and automated management of resources in 5G open RAN (O-RAN) networks, known as xApps, and the underlying algorithms which are based on. The final goal is to be able to take decisions over the network resources, but due to the size and delay caused during data recollection and training for the aim of creating the AI/ML models, these can not be generated online. For that reason, in the form of offline, non-real time training is developed so that in the final deployment the models can be efficiently applied directly in real-time decision making.

Methodology – The steps followed for the elaboration of this work are:

- Research about state of art of 5G O-RAN and its architecture.
- Research about current release in O-RAN Community Software and different repositories (GitHub, Jira, Gerrit, etc.).
- Analysis of the published Traffic Steering use case and the xApps that compose it, as well as study of the associated literature.
- Understanding the use case within O-RAN architecture and its advantages.
- Investigation about existing published datasets and possible improvements of techniques for training the model using the datasets.
- Analysis and testing of the xApps codes (former and current releases).
- Testing new ML models to improve accuracy in the predictions and general efficiency of the xApps that compose the use case under study.

Prototype development and laboratory work – The codes for this study have been developed following the specifications of 5G O-RAN release, allowing the compatibility for any user that want to test with their own data. These published open-source codes are a result of testing and fixing the current releases, as well as the following evaluation and implementation of different new ML models.

Results – The results obtained are:

- Modifications and improvements to xApps of Traffic Steering use case, as well as study of the benefits of the latter have been studied.
- Adaptation of open-source codes for Dummy evaluation.
- New AI/ML proposals for AD and QP xApp, where Neural Networks achieves good accuracy and appears as a promising algorithm.

- New design proposal for QP xApp using only UE metrics, in order to simplify the algorithm and exchanged data, and predict correctly the throughput.

Future Guidelines – Some future challenges can be the testing of the proposed ML models with bigger and more realistic datasets, as well as the xApps evaluation inside the complete RIC deployment. For this, to lower computational complexity can be necessary to assure near-real time functioning. Also, the study of real-time xApps analytics and how it affects the load and latency network is important for 5G O-RAN deployment.

Publications – Open-source contribution published in GitHub: `https://github.com/douniabb/ad-xapp-dummy` and `https://github.com/douniabb/qp-xapp-dummy`.

Abstract – One of the key enabling elements of the new 5G technology is the virtualization of the traditional hardware components of telecommunication networks. In 5G, traditional dedicated hardware is replaced by generic hardware with software running in the cloud or in the edge (for lower latency), so the core network and interfaces between components increase in importance as the emphasis is on interoperability and accessibility between the different parts to guarantee the service.

As a result of this virtualization, the initiative arises to make the RAN also virtualized, open and interoperable. In this context, O-RAN (Open Radio Access Network) emerges, a radio access network deployment with an open and flexible architecture, whose most characteristic component is the RIC (RAN Intelligent Controller). The latter is designed so that through artificial intelligence (AI) tools managed by dedicated applications, the management of radio resources is facilitated.

In this Master's Thesis, radio resource control applications for O-RAN known as xApps will be analyzed. In particular, the work will focus on the use case of Traffic Steering defined by the O-RAN community, composed of several xApps that deal with Quality of Service (QoS) monitoring, anomaly detection or cell switching. More specifically, different AI/ML algorithms will be evaluated for the optimal performance of xApps for the use case under consideration.

Author: Dounia Boukhayar Bouazza, email: dboubou@teleco.upv.es
Director 1: David Gómez Barquero, email: dagobar@iteam.upv.es
Director 2: Nuria Molner Siurana, email: numolsiu@iteam.upv.es
Submission date: 02-09-22

Contents

1 Introduction

Nowadays, society has adapted to a constant consumption of media, applications and other technology services which in turn has led to the need of enterprises to keep upgrading their contents to beat the market competition. A clear example are media streaming services, such as Netflix, HBO or Prime Video. Every year more and more new streaming platforms are released, which leads to all the other providers to have to upgrade in order to offer more content or enhance their Quality of Experience (QoE). This irrevocably results in an exponential increase of exchanged data through the network, which can not be supported by traditional deployments.

In that context, the latest implemented mobile network technology was the 4th Generation (4G) or LTE which meant a big change of perspective in telecommunications from being simple voice services networks to reaching higher throughput up to 150 Mbps for other data services exchange. However, the flexibility and the virtualization needed for the deployment of new technologies, such as Internet of Things (IoT), has led to the emerge of the New Radio - 5G technology.

The virtualization will allow for service providers to migrate their hardware components to virtualized software in the cloud. With 5G, this is possible thanks to its characteristics of very low latency known as real-time, more security and higher throughput. In that way, all the constant flow of generated data and services can be used in an interoperable, flexible network.

On the other hand, a rise in a multi-vendor environment perspective is developing. The mobile network market has been a closed system, especially in the radio side. A future telecommunication infrastructure where different operators, small or big, could use every component could highly benefit all parts involved with transparency, cost-efficiency and high flexibility.

In order to efficiently manage this new network approach, an open radio access technology has been created, known as *O-RAN* (open RAN). One of its main contributions to upgrade 5G deployment is the addition of AI/ML (Artificial Intelligence / Machine Learning) algorithms that will be able of efficiently managing radio resources in real time following the concept of Self-Organizing Networks (SON) [1]. Consequently, this will all at once deal with the management of huge amount of exchanged data in an autonomous way, as well as the interoperability between different vendor components, resulting in a fast, scalable and dynamic architecture [2].

Initially launched by 5 major mobile carriers (AT&T, China Mobile, Deutsche Telekom, NTT DOCOMO and Orange) in 2018, O-RAN Alliance [3] it is nowadays supported by over 320 companies and 31 mobile operators around the world, representing how operators, suppliers and researchers can constructively collaborate. Their ecosystem is divided in three main groups: the O-RAN Specification Effort (also divided in several Work Groups), O-RAN Software Community and Testing & Integration. The latter allows to show demonstrations of the technology to introduce its capabilities, even though to this day software and infrastructure are still under development. Since the main goal is to achieve openness and transparency, all work is open to contributors. It is expected that by 2026, open RAN will take over around 15% of the RAN market [4].

2 State of Art

To understand the actual need and how O-RAN has been created, it is important to revise an overview of Radio Access Networks (RAN) evolution [2].

The predominant mobile network generation until this day was *LTE* or *4G*. It is the successor of 3G, also known as UMTS (Universal Mobile Telecommunications System), and it was created for the purpose of improving throughput for multimedia services. To achieve this new approach, the network was modified to work as packet switching instead of circuit switching. Nevertheless, LTE could also be deployed in previous generation structures, being called in this case *nonStandalone* (NSA), or *Standalone* (SA) if it was a new 4G network deployment.

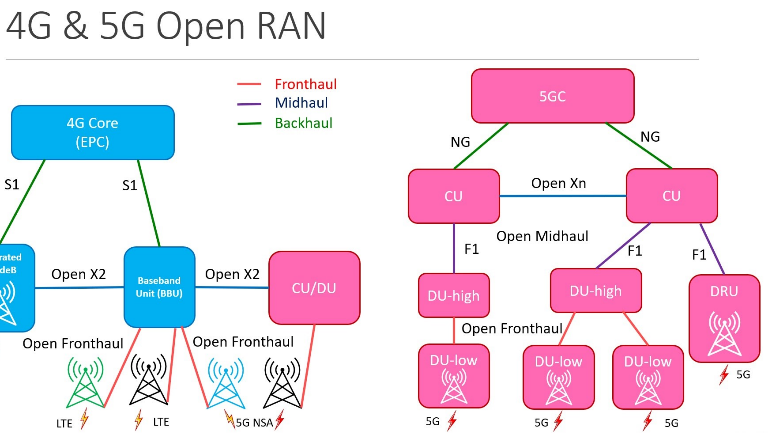


Figure 1: 4G and 5G Architecture[5].

In any case, the LTE architecture is based on a fundamental system called *EPS* (Evolved Packet System) which is formed by a core known as *EPC*(Evolved Packet Core) and by a *RAN* or *EUTRAN* (Evolved UMTS RAN), integrated by a unique logic node type called *eNodeB*. These nodes act as base stations connecting the UEs (User Equipments) with the core network. Meanwhile, the EPC is formed by different components that allow the monitoring of radio resources to improve the QoS (Quality of Service), which is a big addition compared to previous generations.

The issue with LTE as a new telecommunication standard was that it could not reach the desired transfer rate of 1 Gbps. For this, aggregation frequency with *MIMO* (Multiple Input Multiple Output) implementation was developed resulting in *LTE Advanced*.

As stated in the Introduction, the increase of traffic nowadays as a result of a high consumption of multimedia, applications such as social networks, or the rise of IoT technology, has led to issues with network congestion. This implies a need for virtualizing the networks services, and hence the creation a of a new cellular network: *5G NR* (New Radio). The 5G virtualization meant essentially the migration of core operations to the cloud (or edge), resulting that besides achieving a very low latency, there is a better interoperability for interfaces between components. To achieve all this new benefits, other important additions were made to this generation, such as the usage of more antennas with massive MIMO, beam-tracking, scalable OFDM numerology, autonomous flexible slot structure and advanced channel coding.

The new specifications mean that the 5G NR architecture suffered some changes from LTE (see Fig. 1). The main modifications are done to the radio side, known as *NG-RAN*, where a new midhaul is included so that there is a CU (Central Unit) and a DU (Distributed Unit) for each node now called *gNB*. The 4G core is upgraded to a new one called *NGCNA* (Next Generation Core Network) or *5G Core* (5GC), however there is the possibility of maintaining the LTE core as a 5G non-Standalone deployment. For this latter case, *NG-eNB* nodes are used to allow the compatibility between 5GC and EPC.

The core virtualization led to the idea of also virtualizing the radio side, creating the concept of *vRAN* (virtualized RAN). 3GPP (3rd Generation Partnership Project) introduced the DU and CU elements as the evolution path toward vRAN, where several DUs could be merged into a single CU for processing resulting in cost reduction. However, since the RAN components are proprietary, the vRAN can not get its maximum efficiency. In order to deploy a multi-vendor environment and get advantage of the flexibility and intelligence that a virtual RAN offers, the system needs to break from the vendor lock-in design [6].

This emerges the so-called *open RAN* (*O-RAN*), which is a reshape of 3GPP 5G structure with interoperable hardware and open software. O-RAN is a technology with great focus currently for every telecommunication operator, because due to its cost-efficiency, flexibility, transparency and intelligence characteristics it is seen as the future of mobile network market. It essentially consists on the addition of open interfaces and open-source software to the vRAN model. Some of the deployments that are already in development are O-RAN Alliance, Cisco's Open vRAN Ecosystem or Telecom Infra Project's Open RAN MoU Group.

In this work, the study focus is on the O-RAN Alliance [3], which is the trending alliance of operators and providers to define an open and intelligent RAN by proposing specifications and an architecture with building blocks that are complementary to 3GPP standard. The project offers the possibility to mobile operators to independently define and design hardware, which will be open, while using intelligent management tools from an open-source software repository. The main component of O-RAN that aims to achieve these goals is the *RIC* (RAN Intelligent Controller), where apps are allocated with the intelligence that is needed to manage its resources by AI/ML algorithms. The main components of O-RAN architecture are described in more depth in the next chapter.

3 O-RAN Use Case

3.1 O-RAN

In short, O-RAN architecture is based on disaggregated virtualized building blocks and generic hardware build on common standards interconnected by open interfaces and controlled autonomously by open-source software, so that an efficient and interoperable ecosystem of different providers and operators components can take place.

The novel component of O-RAN is the ***RAN Intelligent Controller*** (RIC) which aims to bring into RAN deployments much better performance and efficiency, currently under study. This element helps to add intelligence like a Virtual Network Function (VNF) to the traditional 3GPP components, so that a ***Near-RT RIC*** (Near-Real-Time) is allocated in the radio side, while a ***Non-RT RIC*** (Non-Real-Time RIC) is also included in the management side. The characteristic 5G elements such as the CU and DU are integrated as open functions, as it can be observed in Fig. 2. The O-CU is respectively split into *O-CU-CP* (Control Plane) and *O-CU-UP* (User Plane), where an example of O-RAN virtualization and flexibility is manifested: the control functions can be migrated to the cloud while the essential real time functions for the user are implemented on the generic hardware [2]. Furthermore, the main difference in this radio side, besides the RIC addition, is that the 5G DU is now fragmented in an *O-DU* (O-RAN DU) and an *O-RU* (O-RAN Radio Unit), both interconnected through a new open fronthaul (*Open FH*) [7] [6].

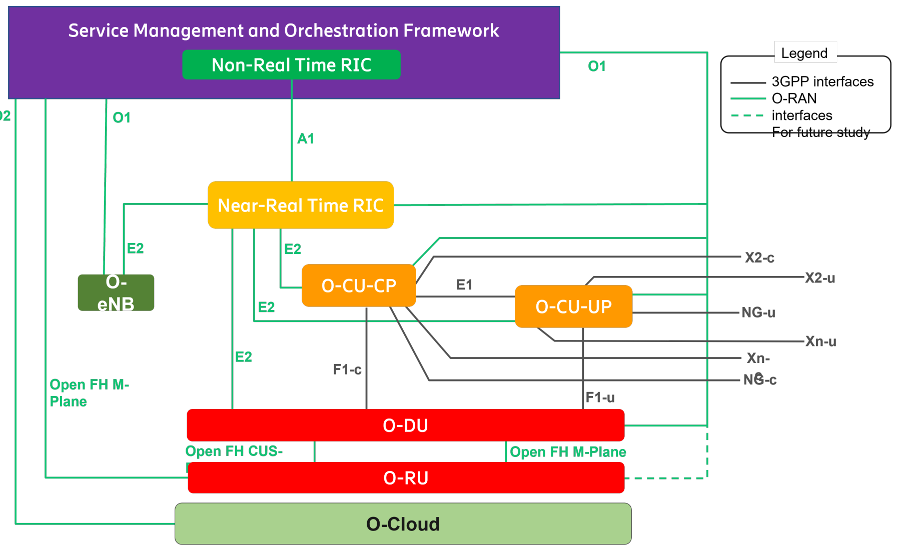


Figure 2: O-RAN Architecture overview. [7]

Now that it has been described how O-RAN has enhanced the 3GPP architecture, the main logical functions of the O-RAN Alliance are explained next in more detail to understand their contributions.

- **Service Management and Orchestration Framework (SMO)**: As its name declares, SMO Framework is the upper O-RAN component that connects to all O-RAN functions through *O1* interface for the system general administration and automation. As an exception, O1 interface can not directly connect the SMO to the O-RU, but instead an *Open Fronthaul M-Plane* interface is deployed since the O-RU virtualization is still in development. The framework comprises different functions to be in charge of inventory, configuration or design, where the non-RT RIC is the main component. On the other hand, through *O2* interface, SMO can communicate with *O-Cloud*, which is the cloud platform where the logical functions of the physical radio nodes are allocated.

- **Non-RT RIC**: Logical component of SMO in charge of non-real time (above 1 second) control of RAN elements to enhance optimization. Its brain are the *rApps*, tools programmed for loading and training AI/ML models offline or useful actions to control the interfaces, making use of the data collected by SMO. It is also in charge of orchestration and the establishment of policies for the models in the radio side through an (*A1*) interface that directly interconnects non-RT and near-RT RIC.

- **Near-RT RIC**: It is the logical element included in the radio side, hence, the near-real time attribution (in the order of ms). It is connected to the lower components through the *E2* interface, which is a very important O-RAN interface due to its responsibility for data acquisition and the delivery of control and optimization decisions to gNB nodes. The actions are decided by ***xApps***. These apps are mainly founded on artificial intelligence and machine learning (AI/ML) algorithms applied to real data gathered from user equipments (UE) and base stations (BS). The algorithms learn to manage radio resources for a big range of goals, such as detection of anomalies, failure prediction and taking immediate decisions.

In this thesis, the main focus will reside in xApps which in essence are AI/ML algorithms for the management of radio resources. The study will be possible thanks to the contributions of *O-RAN Software Community (OSC)*, an O-RAN and Linux Foundation alliance to work and develop on open-source software for this promising future deployment [8]. OSC leads the specifications of the O-RAN building blocks deployment such as the RICs, the xApps and the possible use cases [9]. The different plannings and advances in the project are structured into releases: the first release was in November 2019, *"E"* release is the release that was active through the development of this study and a new release *"F"* was published as on June 2022. Moreover, the current issues and improvements from different developers can be found in many repositories open to the public, such as Gerrit [10] or Jira [11], which can contribute with the help of documentation updated by OSC in [12].

3.2 Use Case and xApps Overview

The main O-RAN use case that is under study to this day by different operators and academia researchers is the Traffic Steering use case (also known as Anomaly Detection use case) [13][14]. The objective is simple: to autonomously detect anomalies present in users in real time, so that the system can quickly ask for a handover to a cell with better quality for the UE.

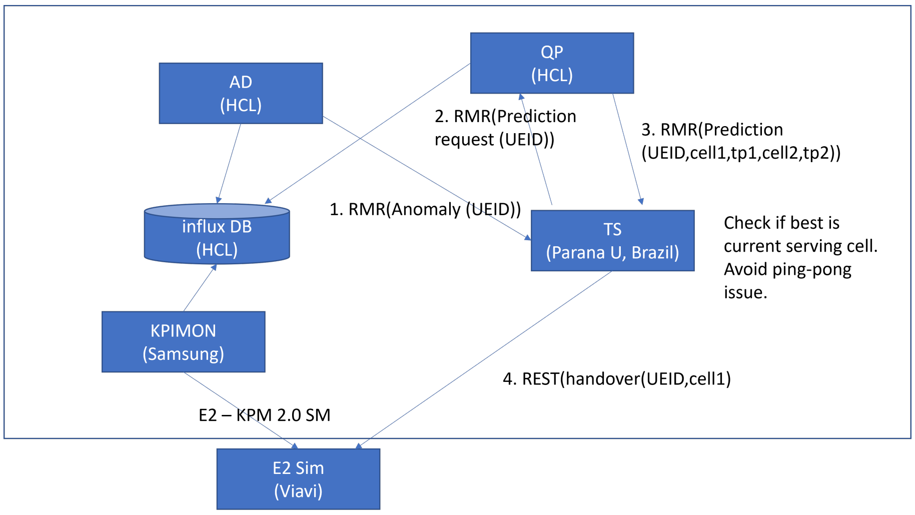

QP
(HCL)
AD
(HCL)
2. RMR(Prediction request (UEID))
3. RMR(Prediction (UEID,cell1,tp1,cell2,tp2))
influx DB
(HCL)
1. RMR(Anomaly (UEID))
TS
(Parana U, Brazil)
Check if best is current serving cell. Avoid ping-pong issue.
KPIMON
(Samsung)
4. REST(handover(UEID,cell1)
E2 – KPM 2.0 SM
E2 Sim
(Viavi)

Figure 3: Traffic Steering Use Case workflow [13].

Ideally, for the near-real-time functioning deployment, the xApps take the cell and UE measurement reports extracted by E2 nodes, which are then transported to a database in the near-RT RIC. As for OSC current trials, the O-RAN mobile traffic is simulated by an E2 Simulator (launched by Viavi) and then, KPIs (Key Performance Indicators) are extracted with *KPIMON* (yet to be implemented), to finally be sent to be stored in a server database (*influx DB*). This latter worfklow is the one represented in Fig. 3.

Once all data is stored in the database, the xApps can extract from it the necessarily information for the use case which comprises the following message exchange:

- **Anomaly Detection (AD)**: Application created for the goal of network optimization when anomalies are found in any of the users. It also identifies the type of anomaly.
- **Traffic Steering (TS)**: Module that gives name to this use case, since it is the one that decides the UE handover to a better cell. AD sends to TS the anomalous UE-ID (User Equipment Identity), and then TS asks for a load prediction request on the serving and neighbor cells of the UE before deciding handover according to A1 policy.
- **QoE Predictor (QP)**: Model that receives TS prediction request of the anomalous UE, and by processing its ID, it obtains its UE and cell-related metrics. When throughput prediction is obtained for every cell, the results are sent back to TS.

3.3 Current Issues

In literature, it can be observed that the main focus of the operators and researchers until this day is on the deployment of the near-RT RIC and non-RT RIC, leaving aside the study and enhancement of the AI/ML algorithms that comprise the xApps. The open-source codes proposed by OSC are slightly modified very slowly, having most xApps the same basic algorithms since the very first release. Since these intelligent tools are the key pillars to achieve the openness and intelligence desired for O-RAN deployment, it is fundamental to correctly develop its functionality.

Very few studies have been recently published in literature for the use of AI/ML algorithms for O-RAN, like [1] or [15], but none are about the enhancement of this specific use case xApps and respective models. This sets a setback for the project unification and progress.

In the next chapter, the analysis of this use case xApps models investigated in this work is detailed, where modifications and new ML models will be evaluated, with the aim of improving some general issues and challenges found on OSC repositories:

- Incorrect evaluation of AI/ML models by the absence of training and testing split of dataset. The models are trained taking into account the whole dataset, by inaccurately predicting values since the intelligence does not predict anything non-apprehended. No sign of cross-validation is identified in the codes.

- Each xApp uses different datasets with different structure and parameters so this does not reflect how the use case could work as a unified system.

- The data under training is very simple to correctly evaluate the model performance. In real context and for future work, datasets gathering more events are needed to train them.

4 Analysis of AI/ML Algorithms for xApps

4.1 Evaluation Scheme

For future 5G O-RAN deployments, when big streams of radio data need to be managed by the RIC, it will be fundamental that machine learning (ML) training takes place in Non-RT RIC since time-sensitive actions have to use models trained beforehand with a set of initialized tuned parameters. Completely untrained models cannot be directly deployed in the network without prior training and testing because it is a time-consuming task that needs to learn how the radio system works and fails, in order to be able to immediately make a decision and take action for an online setting [15][1].

Following the design principles of the use case described in the previous chapter, for the xApps analysis, an evaluation scheme is presented in Fig. 4. Different AI/ML algorithms are proposed for the training of the chosen models according to the functionality of each xApp [16]. For this, the idea of the evaluation scheme is to directly use a unique database for the UE and Cell reports, obtained from simulated E2 nodes traffic.

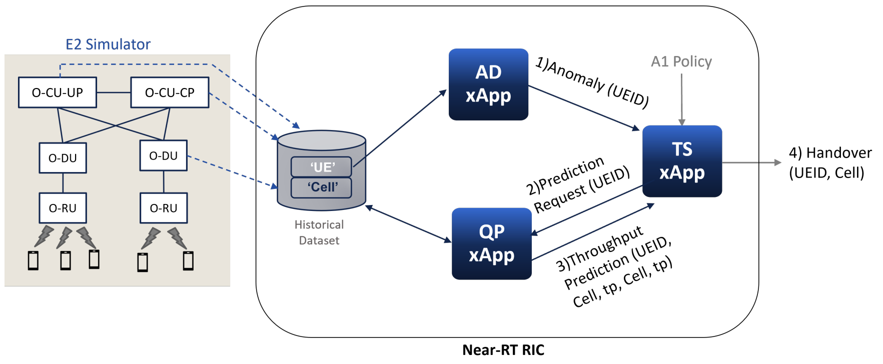


Figure 4: xApps Evaluation Scheme.

Since *influxDB* server is still being fixed for its implementation within the xApps deployment, a "DUMMY" database is used instead for the development of the xApps codes. For this reason, in our evaluation scheme, the data collection is represented directly as the Historical Database for ML training (historical data) and testing (will emulate live radio measurement). The dummy data has been obtained from public OSC repositories for each xApp, but for the sake of an objectified evaluation and the validation of the complete system, the same dataset will be used for all models.

Under that same dataset principle, the summary of the xApps evaluation for different ML models consists on:

1. **Training**: comprises the historical data collection and preprocessing plus the AI/ML model training as if it was non-RT RIC. It is the point of view taken for this chapter.

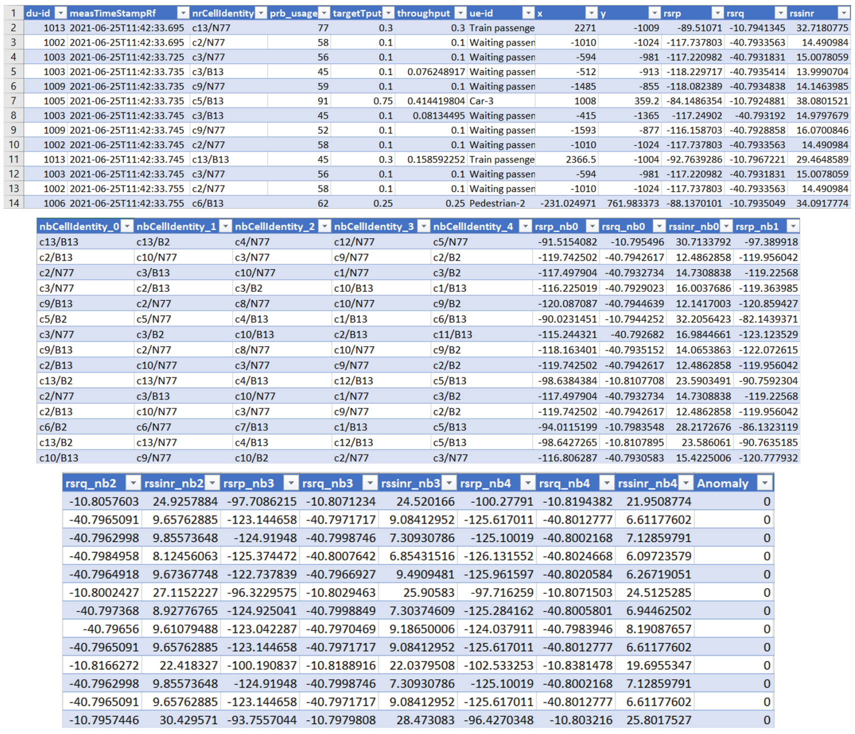
2. **Testing**: emulates the following real metrics acquisition from radio nodes and its immediate decision taking with xApp in near-RT RIC. The latter in this use case will act as the validation part of our AI/ML algorithms analysis.

4.2 Dataset

The dataset that will be used for the analysis and evaluation of the whole use case is 'valid.csv' (check Fig. 5), included as training and testing data in the last release of Anomaly Detection (AD) xApp from the official O-RAN repository [17].

Main characteristics of this simulated data:

- 10000 samples
- Labelled with Anomaly {0,1}
- DU-ID : {1001-1006, 1009, 1010, 1013}
- Timestamp Measure : {11:42:33.695 – 11:43:06.815} every 10 ms
- 9 Cells: {c1 – c6, c9, c10, c13}
- UE-ID: {Car, Waiting passenger, Train passenger}
- Signal strength parameters: {[RSRP, RSRQ, RSSINR], 'PRB usage', 'Throughput'}
- Five neighbor Cells {0-4}
- Anomalies {2568 / 10000 = 0.257}



1	du-id	measTimeStampRf	nrCellIdentity	prb_usage	targetTput	throughput	ue-id	x	y	rsrp	rsrq	rssinr
2	1013	2021-06-25T11:42:33.695	c13/N77	77	0.3	0.3	Train passenge	2271	-1009	-89.51071	-10.7941345	32.7180775
3	1002	2021-06-25T11:42:33.695	c2/N77	58	0.1	0.1	Waiting passen	-1010	-1024	-117.737803	-40.7933563	14.490984
4	1003	2021-06-25T11:42:33.725	c3/N77	56	0.1	0.1	Waiting passen	-594	-981	-117.220982	-40.7931831	15.0078059
5	1003	2021-06-25T11:42:33.735	c3/B13	45	0.1	0.076248917	Waiting passen	-512	-913	-118.229717	-40.7935414	13.9990704
6	1009	2021-06-25T11:42:33.735	c9/N77	59	0.1	0.1	Waiting passen	-1485	-855	-118.082389	-40.7934838	14.1463985
7	1005	2021-06-25T11:42:33.735	c5/B13	91	0.75	0.414419804	Car-3	1008	359.2	-84.1486354	-10.7924881	38.0801521
8	1003	2021-06-25T11:42:33.745	c3/B13	45	0.1	0.08134495	Waiting passen	-415	-1365	-117.24902	-40.793192	14.9797679
9	1009	2021-06-25T11:42:33.745	c9/N77	52	0.1	0.1	Waiting passen	-1593	-877	-116.158703	-40.7928858	16.0700846
10	1002	2021-06-25T11:42:33.745	c2/N77	58	0.1	0.1	Waiting passen	-1010	-1024	-117.737803	-40.7933563	14.490984
11	1013	2021-06-25T11:42:33.745	c13/B13	45	0.3	0.158592252	Train passenge	2366.5	-1004	-92.7639286	-10.7967221	29.4648589
12	1003	2021-06-25T11:42:33.745	c3/N77	56	0.1	0.1	Waiting passen	-594	-981	-117.220982	-40.7931831	15.0078059
13	1002	2021-06-25T11:42:33.755	c2/N77	58	0.1	0.1	Waiting passen	-1010	-1024	-117.737803	-40.7933563	14.490984
14	1006	2021-06-25T11:42:33.755	c6/B13	62	0.25	0.25	Pedestrian-2	-231.024971	761.983373	-88.1370101	-10.7935049	34.0917774

nbCellIdentity_0	nbCellIdentity_1	nbCellIdentity_2	nbCellIdentity_3	nbCellIdentity_4	rsrp_nb0	rsrq_nb0	rssinr_nb0	rsrp_nb1
c13/B13	c13/B2	c4/N77	c12/N77	c5/N77	-91.5154082	-10.795496	30.7133792	-97.389918
c2/B13	c10/N77	c3/N77	c9/N77	c2/B2	-119.742502	-40.7942617	12.4862858	-119.956042
c2/N77	c3/B13	c10/N77	c1/N77	c3/B2	-117.497904	-40.7932734	14.7308838	-119.22568
c3/N77	c2/B13	c3/B2	c10/B13	c1/B13	-116.225019	-40.7929023	16.0037686	-119.363985
c9/B13	c2/N77	c8/N77	c10/N77	c9/B2	-120.087087	-40.7944639	12.1417003	-120.859427
c5/B2	c5/N77	c4/B13	c1/B13	c6/B13	-90.0231451	-10.7944252	32.2056423	-82.1439371
c3/N77	c3/B2	c10/B13	c2/B13	c11/B13	-115.244321	-40.792682	16.9844661	-123.123529
c9/B13	c2/N77	c8/N77	c10/N77	c9/B2	-118.163401	-40.7935152	14.0653863	-122.072615
c2/B13	c10/N77	c3/N77	c9/N77	c2/B2	-119.742502	-40.7942617	12.4862858	-119.956042
c13/B2	c13/N77	c4/B13	c12/B13	c5/B13	-98.6384384	-10.8107708	23.5903491	-90.7592304
c2/N77	c3/B13	c10/N77	c1/N77	c3/B2	-117.497904	-40.7932734	14.7308838	-119.22568
c2/B13	c10/N77	c3/N77	c9/N77	c2/B2	-119.742502	-40.7942617	12.4862858	-119.956042
c6/B2	c6/N77	c7/B13	c1/B13	c5/B13	-94.0115199	-10.7983548	28.2172676	-86.1323119
c13/B2	c13/N77	c4/B13	c12/B13	c5/B13	-98.6427265	-10.8107895	23.586061	-90.7635185
c10/B13	c9/N77	c10/B2	c2/N77	c3/N77	-116.806287	-40.7930583	15.4225006	-120.777932

rsrq_nb2	rssinr_nb2	rsrp_nb3	rsrq_nb3	rssinr_nb3	rsrp_nb4	rsrq_nb4	rssinr_nb4	Anomaly
-10.8057603	24.9257884	-97.7086215	-10.8071234	24.520166	-100.27791	-10.8194382	21.9508774	0
-40.7965091	9.65762885	-123.144658	-40.7971717	9.08412952	-125.617011	-40.8012777	6.61177602	0
-40.7962998	9.85573648	-124.91948	-40.7998746	7.30930786	-125.10019	-40.8002168	7.12859791	0
-40.7984958	8.12456063	-125.374472	-40.8007642	6.85431516	-126.131552	-40.8024668	6.09723579	0
-40.7964918	9.67367748	-122.737839	-40.7966927	9.4909481	-125.961597	-40.8020584	6.26719051	0
-10.8002427	27.1152227	-96.3229575	-10.8029463	25.90583	-97.716259	-10.8071503	24.5125285	0
-40.797368	8.92776765	-124.925041	-40.7998849	7.30374609	-125.284162	-40.8005801	6.94462502	0
-40.79656	9.61079488	-123.042287	-40.7970469	9.18650006	-124.037911	-40.7983946	8.19087657	0
-40.7965091	9.65762885	-123.144658	-40.7971717	9.08412952	-125.617011	-40.8012777	6.61177602	0
-10.8166272	22.418327	-100.190837	-10.8188916	22.0379508	-102.533253	-10.8381478	19.6955347	0
-40.7962998	9.85573648	-124.91948	-40.7998746	7.30930786	-125.10019	-40.8002168	7.12859791	0
-40.7965091	9.65762885	-123.144658	-40.7971717	9.08412952	-125.617011	-40.8012777	6.61177602	0
-10.7957446	30.429571	-93.7557044	-10.7979808	28.473083	-96.4270348	-10.803216	25.8017527	0

Figure 5: Parameters of "valid.csv" dataset.

This dataset belongs to UE namespace, and the main measured metrics for each UE are [18]:

- **Signal strength (RSRP, RSRQ, RS-SINR)** for actual and four neighboring cells (E2 Node is CU-CP).
- **PRB Usage** : Average number of downlink PRBs (Physical Resource Block) used by UE. It is cell-wide and by UE (E2 Node is DU).
- **Throughput** : Number of average downlink PDCP (Packet Data Convergence Protocol) bytes based on measurement period. Also, cell-wide measured (E2 Node is CU-UP).

4.3 Anomaly Detection (AD) xApp

Anomaly Detection (AD) xApp [19] is the application used to detect anomalies in the users and serving cells in order to optimize the network. The detected anomalous UEs are then communicated to Traffic Steering xApp in order to ask for its optimization, that is, its handover to a better cell.

In the TS use case detailed before, has also been declared that AD is used for populating the database when asking for reports from the simulator or real-time measurements from E2 interface.

Two open-source code releases have been published for this xApp:

- Release 0.0.1 (2020): Using Random Forest Classifier model.
- Release 0.0.2 (2021): Using Isolation Forest model.

In the next subsections, these algorithms and other typical for detecting anomalies will be compared.

4.3.1 O-RAN AD xApp Release

The first step taken into the analysis of this xApp was the research about the first published release (0.0.1) which used Random Forest as a model. The premise of the developed algorithm was to classify every timestamp of UE metrics into a quality Category: Excellent, Good, Average or Poor Signal. The implementation was found not very reliable since the testing is only based on comparing if a predicted Category was obtained in training for the UE-ID.

For that reason, in this demonstration, last release 0.0.2 [17] algorithm is analyzed and tested. The idea in the new release is not to classify the signal in a Category for every 10-ms timestamp, but to detect in a range of time if there are big changes by the learning of Isolation Forest model.

The main steps taken for training and testing, represented in Fig. 6 are:

1. Populate database and extract UE metrics. In this case, our UE metrics for training are used (”valid.csv”).
2. Split dataset into training (0.75) and testing (0.25).

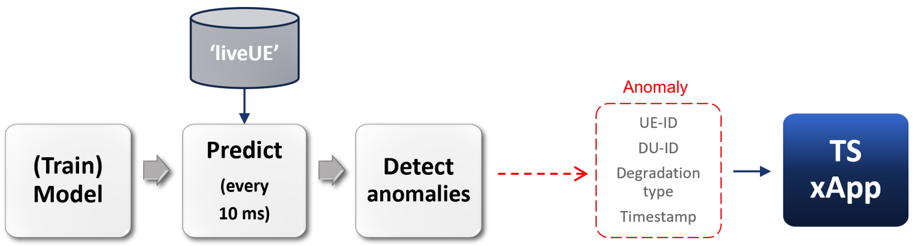


Figure 6: AD xApp workflow.

3. Preprocess data and obtain useful columns: *prb usage*, *throughput*, *rsrq* and *rssinr* (rsrp is dropped due to high correlation). For train data, 'Anomaly' column is dropped, while test data is filtered with only 'Anomaly' column (0 if non-anomalous, 1 if anomalous).

4. Training: Build Isolation Forest model with only the train X data, since is unsupervised learning method. Save fitted model.

5. When the ML model is loaded, a predict function is executed every 10 milliseconds. In live implementation, this would suppose that every 10 ms live measurements are extracted from the database. In this demo, the valid test data (0.25) will be used instead.

6. Inside predict function, the saved IF model is loaded and applied for prediction to the test data. The prediction detects anomalous records and classifies them as -1 or 1, which then is processed as 0 or 1 so that the validation through label comparison can take place.

7. If the input UE-ID is detected as anomalous (1), the UE-ID information is sent to the TS xApp, with the degradation type specified.

In the next subsections, the different models are explained to understand the results.

4.3.2 Classification Models

4.3.2.1 Random Forest

Random Forest (RF) is the default model for Release 0.0.1. It belongs to the supervised learning models family. This algorithm builds decision trees from random samples and takes their majority vote for classification, while in case of regression the average is taken[20][21] .

As shown in Fig. 7, the training observations are separated recursively creating different branches until every observation is isolated in a terminal node. Then, results (predictions) are obtained from each tree and depending on majority votes the best option is chosen as final result. In other words, RF combines predictions from the decision trees and selects the best prediction. The average is taken to improve the predictive accuracy of the model.

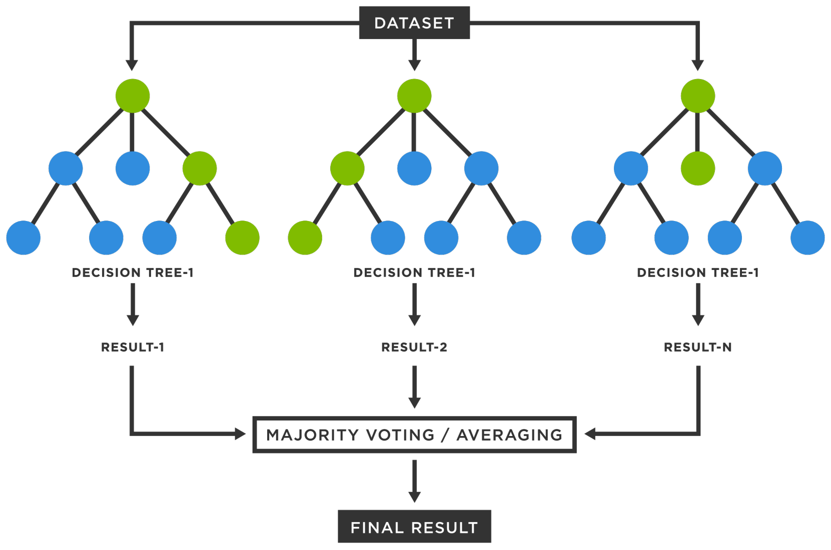


Figure 7: Random Forest decision trees [22].

The algorithm used for Release 0.0.1 is adapted in this work for Release 0.0.2, since it is based on a different dataset and approach:
Release 0.0.1 → Category for signal quality classification (based on some thresholds)
Release 0.0.2 → Binary classification (anomaly)

4.3.2.2 Isolation Forest

Isolation Forest (IF) is another model used for anomaly detection but based on unsupervised learning method. That meaning, it is useful for non-labeled data, when the real classification (anomaly – non-anomaly) of the observations is not known.

It is a method essentially inspired by Random Forest: it is built by the combination of multiple trees called isolation trees, in which the selection of division points is done randomly. Those observations with characteristics different from the rest, will be isolated after a few splits, meaning that the number of nodes necessary to get to those observations since the beginning of the tree (depth) is lower than the rest [23].

To sum up, the steps for the formation of an Isolation Forest are:

1. Establishment of a root node with N training observations.
2. Taking into consideration the observations values, a random "attribute" range q is chosen, from where a random value p is extracted.
3. Division of root node into two new nodes following the $x_q \leq p$ or $x_q > p$ rule.
4. Steps 2 and 3 are iterated until every observation is separated individually into what are called terminal nodes.

For the coding, IF algorithm can be implemented with the packages of Scikit Learn or H2O. The O-RAN implementation uses Scikit [24], where the main parameter to be tuned is the contamination: proportion of expected anomalies in the training data.

The training of the IF model consists on iterating through different contamination values until getting the optimum. Optimum is obtained for the best *f1-score* (error between label and prediction) after the corresponding prediction. That optimum contamination is the one used for the testing and the saved model. In the OSC code, the optimum value of contamination obtained is 0.28, which means that the model has been tuned to detect 28% of anomalies present in input training data. In Section 4.2, it was shown that the dataset includes 26% of anomalies, so it is not a completely accurate model prediction.

4.3.2.3 Neural Networks

A Neural Network design is tested since Deep Neural Networks (DNN) can be very effective for great amounts of data [25], which will be the case in O-RAN real deployments.

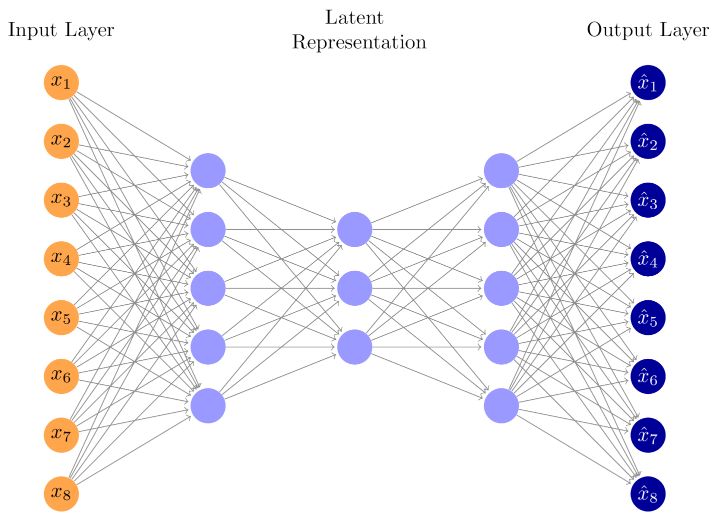


Figure 8: Autoencoder architecture.

First, a simple Sequential Neural Network based on supervised prediction has been developed. A Sequential class is a group of linear stack of layers based on the order in which they are received. The used functions and class belong to the Keras package. The scheme followed for the stacked layers are 4 Dense layers with 2 Dropouts and Adam optimizer.

The designed model yielded an accuracy of 0.9, but it did not improve the learning beyond 30 epochs. For this reason, another possible Neural Network has been investigated for this anomaly detection case: autoencoders [21]. These generative unsupervised deep learning algorithms consist on a neural network formed by an encoder, a bottleneck layer and a decoder respectively. With this structure, autoencoders train for reconstructing data thanks to the latent representation of the input obtained in the bottleneck layer (see Fig. 8).

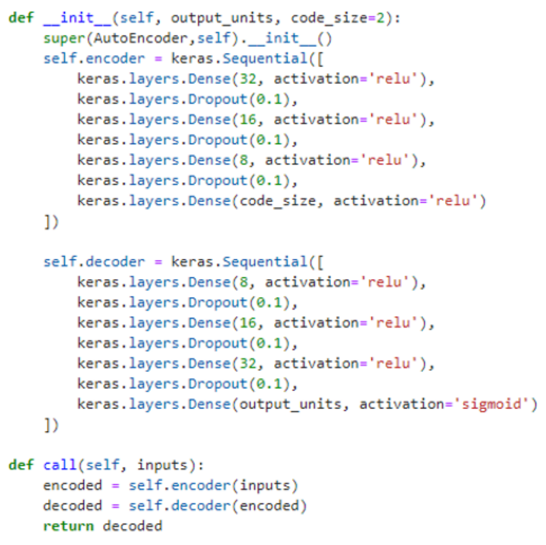
Latent representation means the representation of compressed data in which similar sample points are closer together in space, which is effective for learning the patterns of data features. Another characteristic is that the input size of an encoder structure is bigger than its output size, and vice versa for the decoder.

The autoencoder training consists of trying to minimize the reconstruction loss. The anomalies are identified by checking each sample reconstruction error by the following premises:

- The encoder will receive only normal data as input, and then the bottleneck layer will learn the latent representation of the normal input data.
- The decoder by using the bottleneck layer output, will be able to reconstruct the normal data of the original input.
- An anomaly will be different from a normal sample, meaning that the autoencoder, since it has learned only normal samples, will have trouble reconstructing an anomaly sample resulting in a high reconstruction error.
- A threshold value needs to be specified for the detection of high reconstruction error, hence for detecting an anomalous sample.

Once the autoencoder algorithm is understood, the model and coding for our dataset (”valid.csv”) training and testing are built [26]:

1. Build an adequate Autoencoder model (Fig. 9).



```
def __init__(self, output_units, code_size=2):
    super(AutoEncoder,self).__init__()
    self.encoder = keras.Sequential([
        keras.layers.Dense(32, activation='relu'),
        keras.layers.Dropout(0.1),
        keras.layers.Dense(16, activation='relu'),
        keras.layers.Dropout(0.1),
        keras.layers.Dense(8, activation='relu'),
        keras.layers.Dropout(0.1),
        keras.layers.Dense(code_size, activation='relu')
    ])

    self.decoder = keras.Sequential([
        keras.layers.Dense(8, activation='relu'),
        keras.layers.Dropout(0.1),
        keras.layers.Dense(16, activation='relu'),
        keras.layers.Dropout(0.1),
        keras.layers.Dense(32, activation='relu'),
        keras.layers.Dropout(0.1),
        keras.layers.Dense(output_units, activation='sigmoid')
    ])

def call(self, inputs):
    encoded = self.encoder(inputs)
    decoded = self.decoder(encoded)
    return decoded
```

Figure 9: Autoencoder model design.

2. Normalize data to [0,1]. The scaler used is *MinMaxScaler*.
3. Separate normal samples from anomalous samples in X_train and X_test.

4. Train model.

5. Check Train vs. Validation loss.

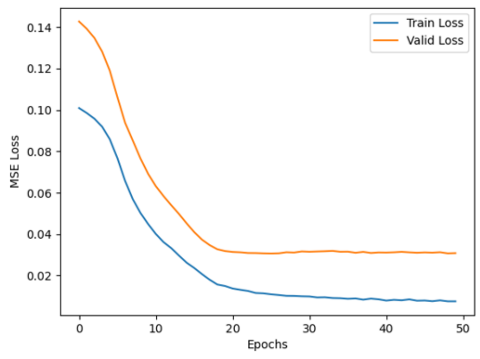


Figure 10: Train vs Validation MSE Loss.

6. Calculate reconstruction loss. (Check Fig. 11).

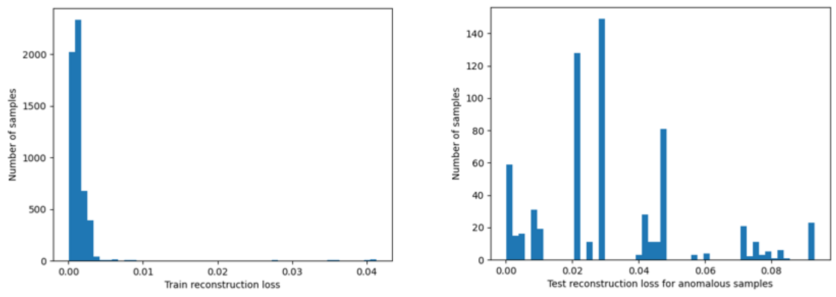


Figure 11: Reconstrucction Loss.

7. Choose a threshold for high reconstruction loss. The threshold is calculated as one standard deviation above the mean.

8. Applying the threshold on testing dataset, predict if a sample is an anomaly. The accuracy obtained is 0.9456.

4.3.3 Results

In the original O-RAN Software Community codes, the Random Forest used on Release 0.0.1 achieved very high accuracy output. For training, the accuracy reached 0.99, while on testing the value was 0.97.

In our modified code based on Release 0.0.2, RF still reaches very high accuracy: 0.98 for training and 0.97 in testing.

As for Isolation Forest, the accuracy obtained in the default Release 0.0.2 algorithm is 0.91. As we can see, the IF accuracy is lower than RF, nonetheless is more adequate for real implementation as stated in literature and other research projects, like [27] where the next conclusion is drawn: being RF a supervised learning, it is capable of learning the behavior of the low complexity dataset simulator. This is understandable knowing how each algorithm works and how they split and build the nodes in the decision trees: Isolation Forest identifies the anomalies while Random Forest only works on the profiling of normal data points.

Isolation Forest works on the basic principle that an outlier data point can be isolated to a node by much lesser number of splits in a tree structure compared to a non anomalous data point. For large amount of data, this can result in much better results than for Random Forest, but with the advantages of increasing learning while more data are possesed can yield that the Autoencoder is the best option. As shown in Table 1, the Autoencoder evaluation for our data gets a reasonable good accuracy reaching 0.95.

	Accuracy	F1-score
RF Train	0.988	0.984
RF Test	0.974	0.966
IF Train	0.913	0.890
IF Test	0.914	0.891
AE Train	0.950	0.899
AE Test	0.946	0.880

Table 1: Accuracy and F1-score for RF, IF and AE.

Other anomaly detection algorithms such as HDBSCAN have also been evaluated. HDBSCAN is a popular unsupervised learning anomaly detection based on clustering, similar to K-mean. It groups data points in clusters and identifies those points which do not belong to a cluster as anomalies (outliers). However, it did not result on good predictions since our dataset is not useful for clustering.

4.4 QoE Predictor (QP) xApp

Once an anomaly is detected, the AD xApp sends the anomalous UE identification to the Traffic Steering (TS) xApp in order to request for its handover to a better quality cell. For the purpose of achieving this enhanced service, the TS xApp needs firstly to know in which of the neighboring cells the user will enjoy in the near future a better Quality of Experience (QoE). In this context, the QoE Predictor (QP) xApp is emerged [28].

The QP xApp is the application that predicts the throughput of the anomalous UE ID requested by the TS xApp, for its current cell and its cell neighbors. The prediction is made based on the signal quality parameters (RSRP, RSRP, SINR) and the cell's load metrics.

The main steps followed for the analysis of this xApp are:

1. Research about current open-source algorithm, which is based on a time series ML predictor, developed by O-RAN Software Community to achieve the objective.

2. Demonstration of the implemented model and observation of obtained results. Specification of modifications made to DUMMY implementation for this evaluation.

3. Analysis and conclusions of the xApp demonstration.

4. According to the research, new algorithm proposal.

4.4.1 O-RAN QP xApp Release

As presented in the AD section, the first step is the demonstration and evaluation of the current algorithm release from the O-RAN open-source community. The last release was published on December 2021 and is known as 0.0.4. [29]. This current algorithm is based on the time series Vector Autoregression model, while previous releases were an attempt of Lineal Regression prediction.

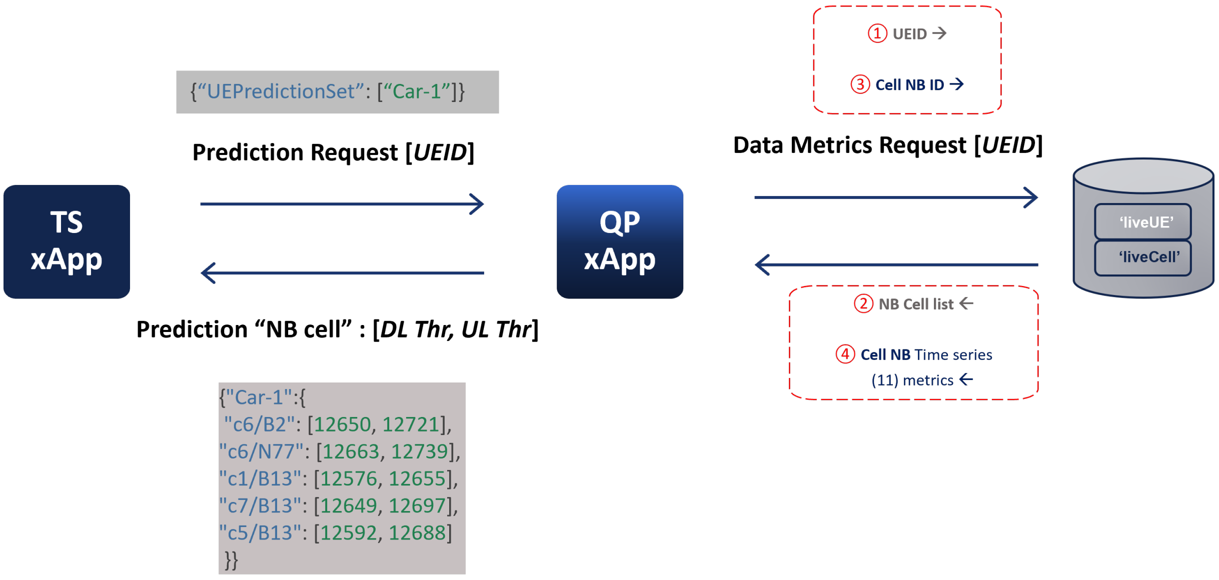

① UEID →
③ Cell NB ID →
{"UEPredictionSet": ["Car-1"]}
Prediction Request [*UEID*]
Data Metrics Request [*UEID*]
TS xApp
QP xApp
'liveUE'
'liveCell'
② NB Cell list ←
④ Cell NB Time series (11) metrics ←
Prediction "NB cell" : [*DL Thr, UL Thr*]
{"Car-1":{
"c6/B2": [12650, 12721],
"c6/N77": [12663, 12739],
"c1/B13": [12576, 12655],
"c7/B13": [12649, 12697],
"c5/B13": [12592, 12688]
}}

Figure 12: QP xApp workflow.

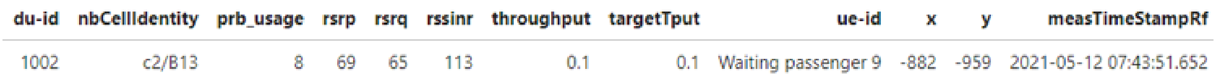
The QP workflow based on a continuous exchange between TS xApp and the database is represented on Fig. 12. In the scheme, we can observe that the expected input is the anomalous UEID, while the expected QP output is a list of downlink and uplink throughput predictions for each cell ID (serving and neighbor cells).

In general terms, when the TS xApp emits a Prediction Request with a specific UEID to QP xApp, the latter immediately asks for Data Metrics Request of the UEID to the database. As explained before, the database is essentially formed by two namespaces: UE Reports (”liveUE”) and Cell Reports (”liveCell”). Ideally, and in real implementation, the reports will be live measurements but for now, and in the dummy implementation, these will be some stored historical data (same as in AD). So, the prime thing that QP needs to acknowledge are the cell list related to the UEID: its serving and neighbor cells. For this, QP request the cell identifications of the UEID to the UE report namespace.

Once QP acknowledges the UEID cell IDs, the algorithm starts a loop for each cell ID to obtain its metrics from each Cell ID Report in the DB. All of these Cell ID Reports are in the form of time series metrics. A Vector Autoregression (VAR) model is performed on each of these Cell time series in order to forecast the next throughput value.

The Dummy implementation has been configured as follows:

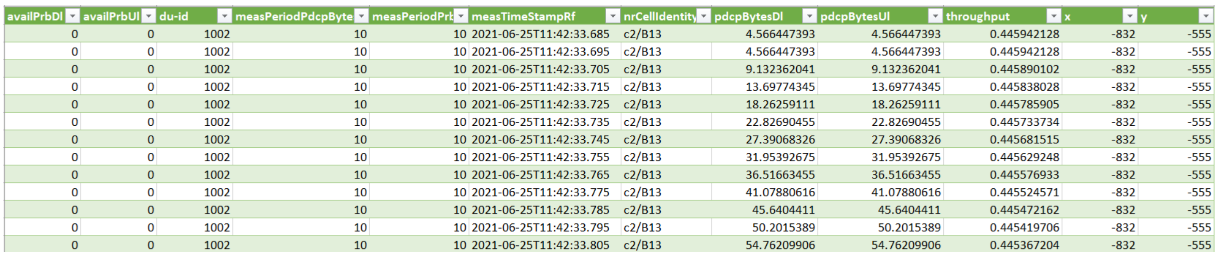
- **"liveUE"** : As UE metrics, only a neighbor cell of a specific supposed "anomalous" UE are specified in the dummy code and shown in Fig. 13. The UE-ID under test is called "Waiting passenger 9", and the neighbor cell identity is "c2/B13".



du-id	nbCellIdentity	prb_usage	rsrp	rsrq	rssinr	throughput	targetTput	ue-id	x	y	measTimeStampRf
1002	c2/B13	8	69	65	113	0.1	0.1	Waiting passenger 9	-882	-959	2021-05-12 07:43:51.652

Figure 13: Dummy UE-ID for QP xApp.

- **"liveCell"** : The data with cell metrics for our cell under test, "c2/B13", are stored as a time series in a file named "dummy.csv".



availPrbDl	availPrbUl	du-id	measPeriodPdcpByte	measPeriodPrk	measTimeStampRf	nrCellIdentity	pdcpBytesDl	pdcpBytesUl	throughput	x	y
0	0	1002	10	10	2021-06-25T11:42:33.685	c2/B13	4.566447393	4.566447393	0.445942128	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.695	c2/B13	4.566447393	4.566447393	0.445942128	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.705	c2/B13	9.132362041	9.132362041	0.445890102	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.715	c2/B13	13.69774345	13.69774345	0.445838028	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.725	c2/B13	18.26259111	18.26259111	0.445785905	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.735	c2/B13	22.82690455	22.82690455	0.445733734	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.745	c2/B13	27.39068326	27.39068326	0.445681515	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.755	c2/B13	31.95392675	31.95392675	0.445629248	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.765	c2/B13	36.51663455	36.51663455	0.445576933	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.775	c2/B13	41.07880616	41.07880616	0.445524571	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.785	c2/B13	45.6404411	45.6404411	0.445472162	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.795	c2/B13	50.2015389	50.2015389	0.445419706	-832	-555
0	0	1002	10	10	2021-06-25T11:42:33.805	c2/B13	54.76209906	54.76209906	0.445367204	-832	-555

Figure 14: Dummy Cell Data "*dummy.csv*".

The columns *pdcpBytesDl* and *pdcpBytesUl* (PDCP, Packet Data Convergence Protocol) are the throughput parameters chosen by the open-source software community which will be used as past values for the VAR model forecasting for each cell.

To put it briefly, Vector AutoRegression models multiple time series as a function of its past values and relaying on the inter-dependencies between the time series, that is their dynamic interrelationship. The VAR model required the variables to be stationary in order to detect their serial correlation. So, in summary, the VAR predictor are learned time delayed values (lags) of the parameters. The number of forecasted observations and the lag order are configurable values [30].

Once all the elements of this xApp are understood, the workflow of the published code is described next in detail as a means of understanding the result:

1. First of all, the xApp needs to populate the database for the training and testing, which in this case are the Dummy metrics shown before.

2. When the anomalous UE-ID is received ("Waiting passenger 9"), it is used as payload for the predict() function.

3. The predict() function extracts the ueid from the payload, and then extracts the cells identities with an external function called cells(). The cell list is obtained from the UE reports, which in this case, for the sake of simplification, was only a neighbor cell: c2/B13.

4. A loop is performed for each identity in the cell list.

5. For the neighbor cell *c2/B13*, the last 11 metrics (about 100 ms of time measurement) from the time series are read from its "liveCell" reports. In this Dummy version, the first 11 rows are read from "dummy.csv". These are saved as the variable *inp*, and will act as the past values.

6. Then, the training of the VAR model for this cell ID is requested by reading the whole file "dummy.csv". The data is processed in order to filter only the desired columns *pdcpBytesDl* and *pdcpBytesUl*. Both time series are checked for stationarity with ADF test, and if necessary, they are made stationary. A time series is stationary when its statistical properties like mean, variance, and autocorrelation don't change over time. This is essential for the correct prediction of VAR because the algorithm needs the time series to not have trends in order to correctly detect patterns. Only then, a model with the VAR module by *statsmodels* is fitted on the processed data with a desired lag order of 10. At last, the model is saved with the cell ID name.

7. To finish, the forecast of the saved past time series (*inp*) is done with the saved model. It is important to remind that the past data also needs to be made stationary before applying the VAR model.

The predicted throughput result is *"Waiting Passenger 9":* $\{"c2/B13" : [50, 50]\}$ and if we check Fig. 14, the next value of pdcpBytes columns after 11 rows matches exactly with our result.

Even though these very precise algorithm example has in theory correctly predicted the value, many elemental issues have been detected in this AI/ML model along its experimentation:

- First of all, the developed code does not effectuate a validation: it uses all the dataset for the model training and then, part of that learned data from the model is predicted. The model is predicting values that is already familiar with, so this does not reflect if it could forecast correctly unknown data. The AI/ML model needs to essentially implement a training and validation split.

- The columns PDCPBytesDL and UL of "dummy.csv" are periodically increasing values, which are not a great emulation of real measures. Consequently, the forecasting for these columns are very easy to predict.

  It does not make sense that the prediction of the xApp consists of a periodically increasing value (of PDCPBytes) since it is not a reflection of real behavior. In real life, measurements will be irregular time series and in that case, the functioning of this algorithm would not apply.

- Since the VAR predictor is based on the knowledge of past values, large and very variate datasets need to be trained in order to get a good prediction in real life for different scenarios.

  The main goal for this release is to improve the model by training larger and better datasets of UE and cell reports. When different signal strength situations can be learned, the intelligence will be better able for forecasting different possible situations. However, there are not enough manageable amount of data that this implementation can learn for a correct prediction, since it would need to continuously be training.

- Lastly, the *pdcpBytes* values seems to be continuously measured from the exact same position in the cell. Since our main goal is to measure the UE's QoE, and the final throughput is strongly related to how far from the base station is, the main report should be taken from the position of the UE to check how it would accurately act there. In this release, the forecasting of the load (pdcpBytes) of each cell as a time series vector it does not take into account other factors such as position, movement, signal quality, etc. of each UE. This seems to be necessary since the handover is needed because an anomaly has been detected, and the cause of this anomaly is related to those factors (signal strength, PRB usage, etc.), and not only as a consequence of variation in the load of PDCP exchanged in the stack of the cell.

After detecting these issues in the code, different solutions have been proposed for QP modification as a means to an enhanced prediction:

- In the first place, adapt and modify DUMMY for a more accurate evaluation of VAR model in dummy.csv, by applying train and test splits for the time series in the database program.
- Test current VAR model prediction for different and most importantly, more complex data.
- Try different time series ML predictors and compare its performance.
- Use metrics from the UE report namespace, which takes into account important related QoE parameters such as position, signal quality metrics, etc.

In order to continue working with our main dataset under test (valid.csv), some modifications to the algorithm have been done and detailed in the next subsection.

4.4.2 Proposal of QP Modification

To keep the line of research of this thesis, the dataset used to test this xApp will be 'valid.csv' which is the .csv file used under test for AD xApp. In that way, an assessment for the whole system result can be made. The main proposed approach is to predict the UE measured throughput instead of the cell wide aggregated values of PDCP bytes.

As specified in Section 4.2, the dataset 'valid.csv' is simulated as data with only UE metrics. Because of this, in that dataset there is no measurement from the cell's layer point of view and, hence, the main parameter analyzed for throughput prediction in the default release for QP xApp is not included in 'valid.csv' (check Fig. 14 and Fig. 5). This parameter is PDCP Bytes for downlink and uplink. So, as a proposal for a new algorithm for QP xApp, the throughput forecasting with only UE metrics is proposed.

One of the big advantages of using the 'throughput' column from "UE" namespace reports instead of the one from "Cell" reports, is the exchange of lesser traffic between xApps and databases since only UE metrics will be asked for, that being for training or for live measuring from E2 interface. Another objective is a more robust prediction because the algorithm will take into account important throughput-related metrics such as the signal strength received from each UE in a specific cell, as opposed to the disadvantages of the actual time series model explained before.

For the sake of clarity, an overview of the main parameters in *figvalid* needs to be observed once more. As represented in the columns, the throughput value for each UE is stated, with the respective related metrics such as position and signal quality (rsrp, rsrq, rssinr). Likewise, for every UE the list of five neighbors cell ID's and their respective signal strength are measured. So the idea for the planned algorithm is to take advantage of all these measurements included in only one dataset (and in real O-RAN implementation, will be taken only from UE database interface).

The first obstacle of 'valid.csv' is the absence of 'throughput' reports for the UE's cell neighbors, needed for the QP prediction. Knowing that the signal strength metrics are related with the throughput value, an AI/ML model that learns the correlation between these signal strength measurements and the throughput needs to be developed. After training this model, it can be applied for the throughput prediction for the signal strength metrics of the cell neighbors.

Throughput prediction based on signal strength metrics.

As stated, the dataset under test includes the UE's throughput in the serving cell, but not the throughput values for the UE in the different cell neighbors. Because of this, the first step in the development of this xApp program, will be to predict the throughput of the cell neighbors according to the measurements that we posses (check Fig. 5), which are the signal strength metrics of the serving and neighbor cells and the correspondent throughput value of the UE in the serving cell.

To obtain the correlation between these parameters a multivariate regression problem is assessed. Different regression models will be evaluated, and the one which obtains the better error will be the one used to predict throughput.

Before defining each ML model, the data needs to be correctly preprocessed and afterwards, the required features for X and the label will be specified:

1. Read dataset ('valid.csv')

    (a) Read non-anomalous data.

   (b) Read whole data.

2. Preprocess X data: drop not useful columns, transform data to bring all parameters in same scale (Normalizer) and drop correlated parameters:
   (a) Parameter 'rsrp' is dropped.
   (b) With parameter 'rsrp'.
3. Split train and test dataset as 0.75/0.25 .

These preprocessing settings have been inspired on the ones used in AD xApp, since QP did not implement preprocessing on its data besides stationarity check. Nevertheless, since this use case does not follow the same goal as anomaly detection, and different regression models will be evaluated, some of the settings need to be tuned.

First of all, considering that the main goal is to obtain a model capable of correctly calculating a throughput based on the correlation between the signal quality and the throughput for the serving cell, it seems possible that the model would not correctly learn if it reads anomalous data. For this reason, both possibilities will be tested: training and prediction with whole dataset or with only non-anomalous data.

On the other hand, one of the basis of cleaning and preprocessing data for machine learning is to check and drop parameters with high correlation with other columns which could cause inference with the model functioning. However, for our purpose, what we expect is that the parameters have high correlation so that they are able to obtain throughput values. In any case, both possibilities, 2.a and 2.b, will also be evaluated.

Once the preprocessing has taken place, our main variables will be the predictor X referring to the signal strength metrics ("prb_usage", "rsrq", "rssinr"), and y as the throughput.

Now, three regression problem models will be tested: Linear Regression, Random Forest Regressor and Artificial Neural Networks.

The accuracy metric that will be used is *R2 score*, where 1 is the maximum accuracy.

4.4.2.1 Linear Regression

Prior to defining and applying the Linear Regression model, linearity between features and label has been evaluated by graphic representation, which are shown in Fig. 15.

At first sight, there is no indication of direct linear correlation between the variables, which means that a linear regression will probably not yield the best results. Nonetheless, the model can be tested.

The Linear Regression used model belongs to the package of *sklearn* of lineal models [31].

$$model = LinearRegression()$$
$$model.fit(X\_train, y\_train)$$

In Table 2, the best accuracy has been obtained for the case where non-anomalous data has been trained and the parameter 'rsrp' has been taken into account as a feature in X set.

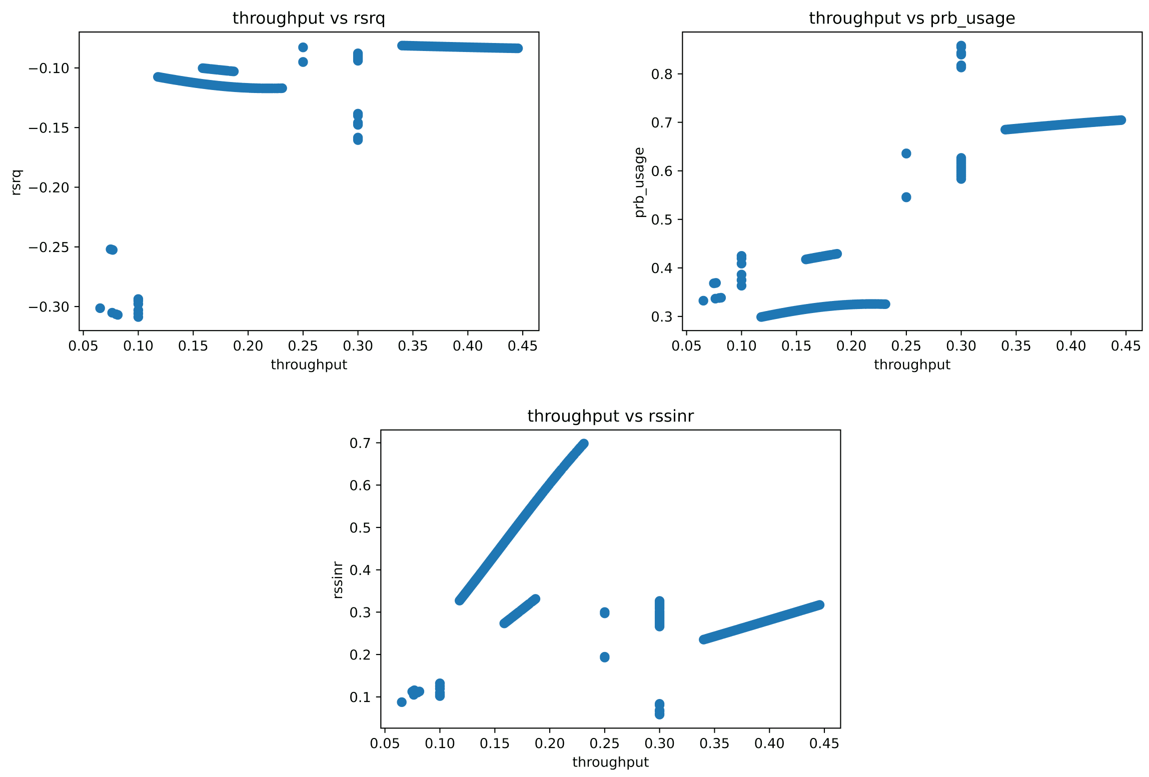


Figure 15: Throughput Label vs. Features [rsrq, prb usage, rssinr].

	1.a	1.b
2.a	0.839	0.497
2.b	0.956	0.609

Table 2: Test Accuracy for Linear Regression.

4.4.2.2 Random Forest Regressor

A Random Forest is also a good regression model to apply to this dataset, especially since it was the default model used beforehand in Anomaly Detection xApp. Besides, it has been shown that the variables are not really related lineally so the random trees can yield great results to detect the correlation in these kind of data.

	1.a	1.b
2.a	0.999	0.865
2.b	0.999	0.868

Table 3: Test Accuracy for Random Forest.

In Table 3, an accuracy of almost one can be observed that has been achieved in the cases of non-anomalous data, for both with or without 'rsrp' column as feature.

4.4.2.3 Artificial Neural Networks

A Deep Neural Network (DNN) for regression will be designed and evaluated. Deep Artificial Neural Networks (ANN) are useful for great amount of data available for learning, so it is not necessary for this use case dataset, but in the future when large amounts of data from the live UE measurement are done, ANN can be more efficient than the former models as explained in 4.3.3. The proposed ANN is constructed with Sequential model by Keras, which can allow to connect a stack of 'Dense' and 'Dropout' layers. Values such as the number of hidden layers or the number of neurons in each one need to be configured and tuned to achieve the maximum accuracy in the model. The used optimizer is 'Adam'.

Furthermore, after defining the sequential model, the batch size and number of epochs need to be tuned because it directly affects the model performance. The accuracy obtained for dataset with only normal measurements and with 'rsrp' column yields an accuracy of 0.92.

To conclude, the regression model which yielded a better accuracy score was Random Forest with a 0.999. For that reason, it is the chosen intelligent model used to predict the throughput values for the neighbor cells of 'valid.csv' dataset. The settings used are filtering and training the model for non-anomalous data and without dropping 'rsrp' column. Nevertheless, it is important to take in mind that in use cases of bigger datasets or live measurement, Neural Network will most probably yield better results than Random Forest.

The input used for the Random Forest Regressor are the signal quality metrics of every neighbor and the characteristic PRB usage of the UE, while the output is a throughput column for each cell neighbor. These new throughput columns are saved into 'valid.csv' dataset in order to use it afterwards as a time series forecasting. For this goal, two different auto-regression models will be evaluated: VAR and ARIMA model.

4.4.3 VAR Model Prediction

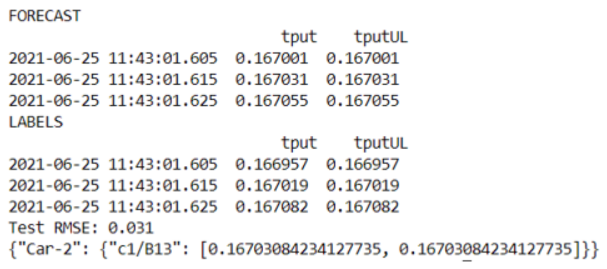
The premise of the Vector Autoregression applied on 'dummy.csv', is now tested for our new extended dataset, called 'valid-tput.csv', for the sake of differentiation from the original data report. To evaluate the time series predictor, the Dummy database program and preprocessing needs to be adapted once again for the purpose of this training and testing workflow.

As for the Dummy code, the new dataset needs to be split into training (0.75) and test (0.25) set, after it is firstly filtered by the UE-ID requested. The training set will be used for the VAR model learning, while the test set will be from where first the UE data will be extracted when is detected as anomalous, and after its timestamp the following data rows are used for the prediction part.

The essential preprocessing function to utilize in this dataset is the interpolation to the time series under test. The file 'valid-tput.csv' is a mix of UE-IDs reports and the measurements are not periodic for every UE, but it consists mostly on irregular time series. In order to apply VAR, it is essential that the time series are equally spaced in time. The time interpolation is applied for every 10 ms to match the O-RAN real-time goal.

The error used for evaluation is *RMSE* (Root-Mean Squared Error), where 0 is minimum error achieved between predicted value and label.

A result example for "Car-2" is shown on Fig. 16, where for its serving cell the throughput for the next timestamp has been predicted and with an error of 0.03.



```
FORECAST
                             tput    tputUL
2021-06-25 11:43:01.605  0.167001  0.167001
2021-06-25 11:43:01.615  0.167031  0.167031
2021-06-25 11:43:01.625  0.167055  0.167055
LABELS
                             tput    tputUL
2021-06-25 11:43:01.605  0.166957  0.166957
2021-06-25 11:43:01.615  0.167019  0.167019
2021-06-25 11:43:01.625  0.167082  0.167082
Test RMSE: 0.031
{"Car-2": {"c1/B13": [0.16703084234127735, 0.16703084234127735]}}
```

Figure 16: Throughput prediction for VAR model and measured error.

4.4.4 ARIMA Model Prediction

ARIMA is an Autoregressive Integrated Moving Average Model, that measures the dependency of a sample with a few of its past observations and with that information, a differencing plus a moving average is applied on the data [32].

In opposition to VAR, this model is about univariate time series which is adequate for our purpose since the only variable to be predicted now is the throughput column, in contrast to the two pdcpBytes columns DL and UL.

The prediction of a value based on the residual error of past values (t-p, or lag) of the time series is based on three parameters p,d,q:

- **p** : lag order, used for autoregression.
- **q** : moving average (MA) order, used as window size.
- **d** : differencing order, to fix stationarity.

For the optimal function of ARIMA, these three parameters need to be tuned. So, how to correctly choose them?

- To select **p**, the lag order, it is necessary to measure the correlation between time series and a certain lag. For this partial autocorrelation (PACF) plot is represented on Fig. 17 with 1st Order Differencing.

  In the graph, the autoregression is measured observing the most significant lag (x axis), which corresponds with p=1.

- For the MA order, **q**, the window size depends on how much of the past is significant for the future. That is, in an autocorrelation (ACF) plot to check the number of lags that cross the correlation threshold, since those lags with high correlations are the ones with higher contribution weight.

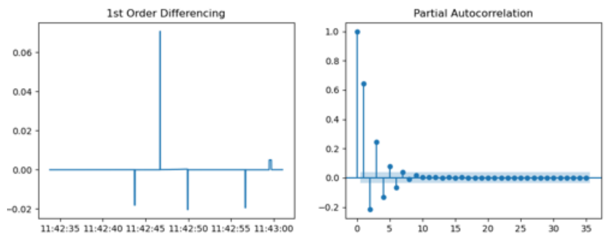


Figure 17: Partial Autocorrelation (PACF) plot vs lag order, with 1st Order Differencing.

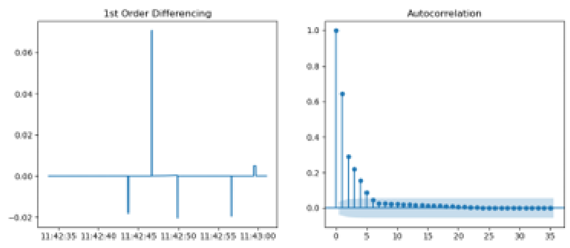


Figure 18: Autocorrelation (ACF) plot vs number of lags.

In Fig. 18, we can observe that the number of lags with higher are correlation are 2.

- To choose **d**, the stationarity of the time series needs to be tested. If it is not stationary, the differencing order iteratively sum one until stationarity is reached.

  There are two ways to check if a time series is stationary: using the Augmented Dickey-Fuller (ADF) test or plotting autocorrelation (ACF) plot with variable differencing order.

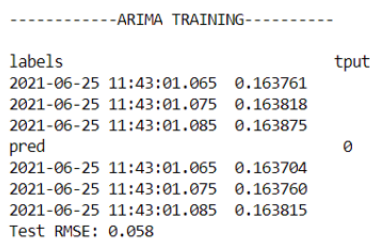
  The ADF test was implemented in the original QP xApp and is based on calculating the p-value. If this value is less than the threshold (0.05), the time series is stationary.

  For our training set, the p-value of the ADF test is shown on Table 4. The threshold is surpassed when d=1, so that becomes the differencing order.

d	p-value
0	0.11
1	4.45e-29

Table 4: ADF test: p-value and differencing order results.

In summary, the ARIMA model to be trained and fitted is the one with parameters (1,1,2).



```
------------ARIMA TRAINING----------

labels                            tput
2021-06-25 11:43:01.065  0.163761
2021-06-25 11:43:01.075  0.163818
2021-06-25 11:43:01.085  0.163875
pred                               0
2021-06-25 11:43:01.065  0.163704
2021-06-25 11:43:01.075  0.163760
2021-06-25 11:43:01.085  0.163815
Test RMSE: 0.058
```

Figure 19: Throughput prediction for ARIMA model and measured error.

4.4.5 Results

The average error obtained for VAR model is 0.031, while for ARIMA it reaches 0.058 for the same anomalous users, as represented in Fig.19. To conclude, VAR model seems to be the best model for throughput prediction in this case, but since there is not much difference between its error and ARIMA's, and taking into account the powerful tuning parameter possibilites of the latter, ARIMA model is a ML algorithm to also take into consideration for other possible dataset complexities.

5 Validation of Proposed AI/ML Scheme

After the analysis elaborated in Section 4, with the selected most efficient AI/ML models for AD and QP xApps, a complete use case validation will be presented. Accordingly, we will follow the evaluation scheme shown in Fig. 4, emulating the testing setup: the ML models are already trained and loaded into Near-RT RIC and a flow of live radio data (*"liveUE"*) is uploading the database.

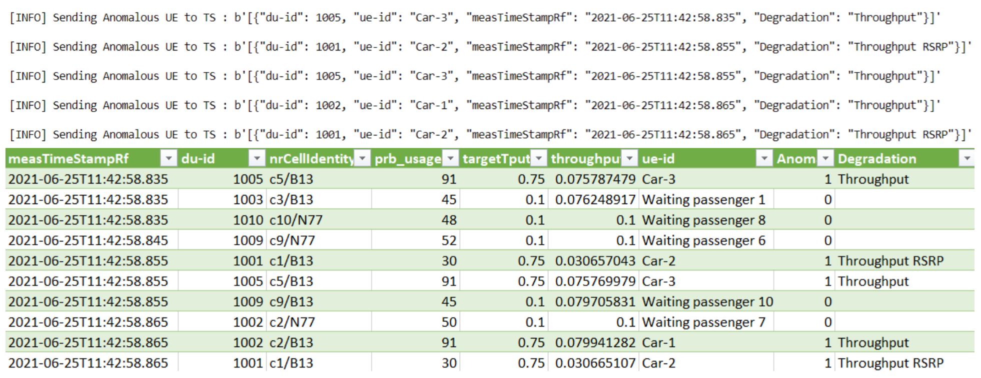
Since the RIC xApp Framework has not been deployed for this work, some modifications are made to the files of the xApp releases in order to start its execution as a Dummy mode. Then, the idea is to check step by step the exchanged messages that the xApps would generate. This will be enough to examine the results of the modified models: for AD xApp, the Autoencoder is implemented as the model, while for QP xApp the modified VAR model to predict throughput using only UE metrics is tested.

In short, the main modifications made to AD xApp codes (compatible with OSC release) are:

- The first file adapted to our purposes is "main.py", where all functions (train, predict and degradation type choice) are executed according to the Dummy premise. After each prediction, it prints on screen the message that AD would send to TS when an anomalous UE is found.

- *DUMMY* section of "database.py" it is modified so that the content of *"valid.csv"* is split into 75% for training the model, associated to a variable "train, while the rest of the data (25%) is used for validation and referred to it as "liveUE". In that way, when executing the predict function it reads the "liveUE" data, which comprises unknown measurements for the ML models.

- The DUMMY class included a "write_anomaly" function that is empty. It is programmed so that the Anomaly and Degradation results are added as columns to the processed radio samples and dumped into a *.csv* file.



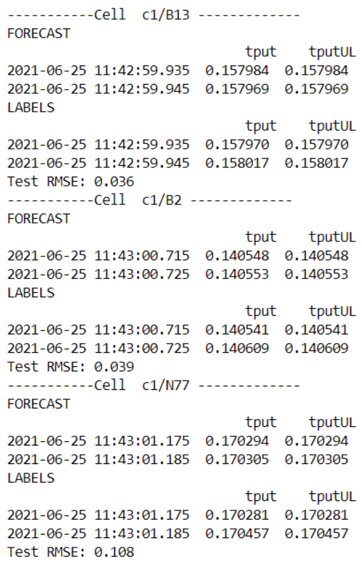
```
[INFO] Sending Anomalous UE to TS : b'[{"du-id": 1005, "ue-id": "Car-3", "measTimeStampRf": "2021-06-25T11:42:58.835", "Degradation": "Throughput"}]'

[INFO] Sending Anomalous UE to TS : b'[{"du-id": 1001, "ue-id": "Car-2", "measTimeStampRf": "2021-06-25T11:42:58.855", "Degradation": "Throughput RSRP"}]'

[INFO] Sending Anomalous UE to TS : b'[{"du-id": 1005, "ue-id": "Car-3", "measTimeStampRf": "2021-06-25T11:42:58.855", "Degradation": "Throughput"}]'

[INFO] Sending Anomalous UE to TS : b'[{"du-id": 1002, "ue-id": "Car-1", "measTimeStampRf": "2021-06-25T11:42:58.865", "Degradation": "Throughput"}]'

[INFO] Sending Anomalous UE to TS : b'[{"du-id": 1001, "ue-id": "Car-2", "measTimeStampRf": "2021-06-25T11:42:58.865", "Degradation": "Throughput RSRP"}]'
```

measTimeStampRf	du-id	nrCellIdentity	prb_usage	targetTput	throughpu	ue-id	Anom	Degradation
2021-06-25T11:42:58.835	1005	c5/B13	91	0.75	0.075787479	Car-3	1	Throughput
2021-06-25T11:42:58.835	1003	c3/B13	45	0.1	0.076248917	Waiting passenger 1	0	
2021-06-25T11:42:58.835	1010	c10/N77	48	0.1	0.1	Waiting passenger 8	0	
2021-06-25T11:42:58.845	1009	c9/N77	52	0.1	0.1	Waiting passenger 6	0	
2021-06-25T11:42:58.855	1001	c1/B13	30	0.75	0.030657043	Car-2	1	Throughput RSRP
2021-06-25T11:42:58.855	1005	c5/B13	91	0.75	0.075769979	Car-3	1	Throughput
2021-06-25T11:42:58.855	1009	c9/B13	45	0.1	0.079705831	Waiting passenger 10	0	
2021-06-25T11:42:58.865	1002	c2/N77	50	0.1	0.1	Waiting passenger 7	0	
2021-06-25T11:42:58.865	1002	c2/B13	91	0.75	0.079941282	Car-1	1	Throughput
2021-06-25T11:42:58.865	1001	c1/B13	30	0.75	0.030665107	Car-2	1	Throughput RSRP

Figure 20: AD xApp results.

The AD xApp output and messages that would be sent to TS are shown in Fig. 20. The accuracy of the predictions yields 0.9.

For QP xApp evaluation, first of all, the Random Forest model ('RF') for the throughput prediction of neighbor cells with signal strength metrics using the "train" data is loaded into the xApp namespace. As for VAR model, it also uses "train" historical metrics, but since it is a specific predictor for each cell behavior, it is trained once the xApp knows the cells that it needs. Then, the models are loaded with their Cell ID name into QP xApp for future use. The workflow represented in Fig.12 is followed, where a specific anomalous user from the same "liveUE" data is chosen and used as input to the model as if it was sent by TS to request its throughput predictions.



```
-----------Cell  c1/B13 -------------
FORECAST
                              tput    tputUL
2021-06-25 11:42:59.935  0.157984  0.157984
2021-06-25 11:42:59.945  0.157969  0.157969
LABELS
                              tput    tputUL
2021-06-25 11:42:59.935  0.157970  0.157970
2021-06-25 11:42:59.945  0.158017  0.158017
Test RMSE: 0.036
-----------Cell  c1/B2 -------------
FORECAST
                              tput    tputUL
2021-06-25 11:43:00.715  0.140548  0.140548
2021-06-25 11:43:00.725  0.140553  0.140553
LABELS
                              tput    tputUL
2021-06-25 11:43:00.715  0.140541  0.140541
2021-06-25 11:43:00.725  0.140609  0.140609
Test RMSE: 0.039
-----------Cell  c1/N77 -------------
FORECAST
                              tput    tputUL
2021-06-25 11:43:01.175  0.170294  0.170294
2021-06-25 11:43:01.185  0.170305  0.170305
LABELS
                              tput    tputUL
2021-06-25 11:43:01.175  0.170281  0.170281
2021-06-25 11:43:01.185  0.170457  0.170457
Test RMSE: 0.108
```

Figure 21: QP xApp results.

1. The selected UE-ID is "Car-2" (from Fig.20). The input ID is specified in Dummy database file.

2. QP request the database for the neighbor (NB) cells identities of the UEID. The NB cell list obtained from the "liveUE" data is: ['c1/B13', 'c1/B2', 'c1/N77', 'c4/B13', 'c5/B13', 'c6/B13'].

3. For each cell ID, its live data is requested to "liveUE" (serving and neighbor cells).

4. The throughput time series is fetched from filtered cell ID test data until getting 11 samples. Interpolation (of 10 ms) is applied to the time series in order to accurately apply VAR model.

The first thing QP xApp will do is to obtain the throughput values of the neighbor cells of historical data recollected from "liveUE", as described in Section 4.4.2.

When applying VAR model to the UE-ID serving cell and the throughput of the neighbors, the algorithm predicts the next throughput value. The range of prediction time can be modifiable, in this case is after 1 second approximately, the equivalent of the next sample after the 11 interpolated samples. As shown in Fig. 21, the algorithm correctly predicts the next throughput values for ”Car-2” with very low error.

When QP sends the results to TS xApp, the latter will follow its A1 policy sent by SMO in order to hand-off the UE to another neighboring cell whose obtained throughput is 5% higher than in current serving cell.

6 Conclusions and Future Work

The conclusions obtained after completing this project are:

- First of all, through the overview of RAN evolution and the description of O-RAN architecture, it has been proved how this initiative is a promising deployment for future telecommunication networks. In comparison with respect to current radio infrastructure, O-RAN is being prepared to provide several benefits such as flexibility, autonomous functioning, real time responsiveness, cost-efficiency and interoperability for a multi-vendor environment. The novel O-RAN component that will make these characteristics possible is the RIC (RAN Intelligent Controller), divided in non-RT (non-Real Time) RIC and near-RT RIC, the latter being where xApps and its AI/ML algorithms are executed.

- It has been described how the Traffic Steering use case could optimize mobile networks management, through the analysis of the xApps use case published by O-RAN Software Community (OSC). The intelligence developed in the xApps is able to detect anomalies in users and predicts its QoE for an autonomous handover to a better cell.

- Modifications and improvements to O-RAN Anomaly Detection (AD) and QoE Predictor (QP) xApps releases have been done to fix the enumerated issues such as erroneous AI/ML training and validation, as well as adaptation for a Dummy evaluation.

- Evaluation of new AI/ML models, where Neural Networks proposals are promising AI/ML for the real O-RAN deployment with big complex flow of data, due to its increasing learning capabilities. For AD xApp, an autoencoder model has been successfully implemented reaching 0.95 accuracy, which surpasses the 0.91 accuracy value of the last model (Isolation Forest) implemented by OSC.

- A new design proposal for QoE Predictor xApp has been presented, where instead of using metrics from each "Cell" report, the *throughput* parameter from within the same "UE" terminal is predicted. This will mean less data exchanged for the near-RT RIC traffic, and results in a much simplified algorithm since all prediction takes place in one file for the UE. For near-real time prediction, and with the case of 5 neighboring cells (plus the serving cell), instead of executing a loop extracting 5 reports for each "Cell" from the database, extracting the cells information from "UE" report will suppose saving up to 80% of data exchanged for each different UE.

As future challenges for Traffic Steering (TS) xApps development and its AI/ML algorithms testing:

- To test the new proposed AI/ML models in a real RIC deployment, where instead of using dummy data, the model can be configured to extract data from a server database, like "influxDB", or also, extract new data from E2 simulators like *Viavi* [33].

- Instead of only using the 'throughput' parameter, make use of the 'Degradation' anomaly type ('RSRP', 'RSRQ', 'RSSINR' or 'PRB usage') predicted by AD xApp and sent from the UE to TS xApp, as the feature to be analyzed in QP xApp to obtain the better cell handover for Traffic Steering.
- Analysis of end-to-end data collection and real-time analytics in near-RT RIC, and how this would accurately affect the load and latency network due to radio resource management complexity.
- To achieve unification of xApps, O-RAN components and software functions, as well as complete its specifications, to be able to reach the standardization of O-RAN.

7 Acknowledgment

First of all, I would like to thank David Gómez-Barquero for giving me the opportunity of making a project in its group. As for the elaboration of the work, I am very thankful to Nuria Molner for all the help and guiding.

As a student away from home, I am not able to put into words how deeply grateful I am for all the daily support, care and love given from what has been my second family this year: Mari Carmen and Alicia. Besides that, I could not have asked for better classmates. The Master has given me personal growth and learning, but also friendships that I will take with me forever. For all the help and shared moments, thank you Nadia, Valeria, Xuban and Luis Manuel. Specially thankful to the latter for the constant encouragement and for being my best g in this narrative arc.

I would also like to express my deepest appreciation to the support given to me from distance. Thank you Aurora, Lucia, Marina and Tugba for being my rocks no matter from where or when. Your infinite trust in me is what has taken me to where I am today. To the rest of my chocus and my teleco girls, I am happy for having you in my life throughout all these years.

At last, but not least, I want to thank my parents for their patience and belief in me along all my academic journey.

References

[1] Anastasios Giannopoulos, Sotirios Spantideas, Nikolaos Kapsalis, Panagiotis Gkonis, Lambros Sarakis, Christos Capsalis, Massimo Vecchio, and Panagiotis Trakadas. "Supporting Intelligence in Disaggregated Open Radio Access Networks: Architectural Principles, AI/ML Workflow, and Use Cases". *IEEE Access*, 10:39580–39595, 2022.

[2] Sameer Kumar Singh, Rohit Singh, and Brijesh Kumbhani. "The Evolution of Radio Access Network Towards Open-RAN: Challenges and Opportunities". In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6, 2020.

[3] O-RAN ALLIANCE e.V. https://www.o-ran.org/. Last accessed June 2022.

[4] Open RAN and vRAN Forecast Revised Upward, According to Dell'Oro Group - Dell'Oro Group. https://www.delloro.com/news/open-ran-and-vran-forecast-revised-upward/. Last accessed August 2022.

[5] The 3G4G Blog: Open RAN Explanation, Videos, White papers and Other Resources. https://blog.3g4g.co.uk/2021/02/open-ran-explanation-videos-white.html.

[6] Andres Garcia-Saavedra and Xavier Costa-Perez. "O-RAN: Disrupting the Virtualized RAN Ecosystem". *IEEE Communications Standards Magazine*, 5(4):96–103, 2021.

[7] O-RAN Project Revision. O-RAN Architecture Overview - oran master documentation. https://docs.o-ran-sc.org/en/latest/architecture/architecture.html. Last accessed August 2022.

[8] O-RAN software. https://www.o-ran.org/software.

[9] O-RAN Software Community - O-RAN SC - Confluence. https://wiki.o-ran-sc.org/.

[10] Gerrit Code Review - ORAN. https://gerrit.o-ran-sc.org/.

[11] ORAN Jira . https://jira.o-ran-sc.org/.

[12] O-RAN SC Release Documentation - oran master documentation. https://docs.o-ran-sc.org/en/latest/.

[13] Anomaly Detection Use Case - RIC Platform - Confluence. https://wiki.o-ran-sc.org/display/RICP/Anomaly+Detection+Use+Case, 2021.

[14] User Guide - ric-app-ts master documentation. https://docs.o-ran-sc.org/projects/o-ran-sc-ric-app-ts/en/latest/user-guide.html. Last accessed July 2022.

[15] Leonardo Bonati, Salvatore D'Oro, Michele Polese, Stefano Basagni, and Tommaso Melodia. "Intelligence and Learning in O-RAN for Data-driven NextG Cellular Networks". *CoRR*, abs/2012.01263, 2020.

[16] Ved P. Kafle, Yusuke Fukushima, Pedro Martinez-Julia, and Takaya Miyazawa. "Consideration On Automation of 5G Network Slicing with Machine Learning". In *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*, pages 1–8, 2018.

[17] GitHub - o-ran-sc/ric-app-ad. https://github.com/o-ran-sc/ric-app-ad/tree/dawn. Last accessed July 2022.

[18] Traffic Steering Use Case xApps - RIC Applications - Confluence. https://wiki.o-ran-sc.org/display/RICA/Traffic+Steering+Use+Case+xAppsl.

[19] AD xApp - ric-app-ad master documentation. https://docs.o-ran-sc.org/projects/o-ran-sc-ric-app-ad/en/latest/index.html. Last accessed June 2022.

[20] sklearn.ensemble.RandomForestClassifier - scikit-learn 1.1.0 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[21] Tzu-Hsuan Lin and Jehn-Ruey Jiang. "Anomaly Detection with Autoencoder and Random Forest". In *2020 International Computer Symposium (ICS)*, pages 96–99, 2020.

[22] What is a Random Forest? — TIBCO Software. https://www.tibco.com/reference-center/what-is-a-random-forest.

[23] Deteccion de anomalias con Isolation Forest y python. https://www.cienciadedatos.net/documentos/py22-deteccion-anomalias-isolation-forest-python.html.

[24] sklearn.ensemble.IsolationForest - scikit-learn 1.1.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.

[25] Anurag Thantharate, Rahul Paropkari, Vijay Walunj, and Cory Beard. "DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks". In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 0762–0767, 2019.

[26] Intro to Autoencoders - TensorFlow Core. https://www.tensorflow.org/tutorials/generative/autoencoder#third_example_anomaly_detection.

[27] Nuria M., Javier R., Jose Luis C., Pietro P., and et al. "iNGENIOUS - ID4.2 Smart NR and NG-RAN IoT designs", January 2022.

[28] QoE Predictor Overview - ric-app-qp master documentation. https://docs.o-ran-sc.org/projects/o-ran-sc-ric-app-qp/en/latest/overview.html. Last accessed July 2022.

[29] GitHub - ric-app-qp. https://github.com/o-ran-sc/ric-app-qp, 2021.

[30] "Vector Autoregressive for Forecasting Time Series — by Sarit Maitra — Towards Data Science". https://towardsdatascience.com/vector-autoregressive-for-forecasting-time-series-a60e6f168c70.

[31] sklearn.linearmodel.LinearRegression - scikit-learn 1.1.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[32] Pierre A. Cholette and Robert Lamy. "Mutivariate ARIMA forecasting of irregular time series". *International Journal of Forecasting*, 2(2):201–216, 1986.

[33] Open RAN, What is it? — O-RAN Architecture, 5G, and Testing Solutions. https://www.viavisolutions.com/en-uk/solutions/open-ran.