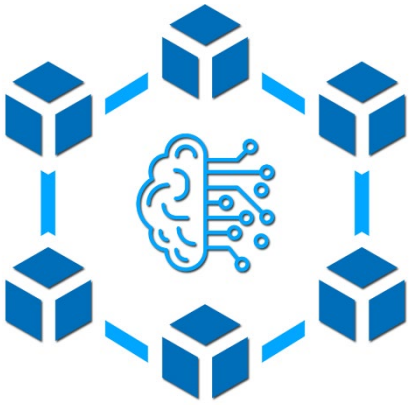

An Artificial Intelligent Aided Unified Network for Secure Beyond 5G Long Term Evolution [GA: 101096456]

Deliverable 2.3

NANCY Network Information Framework

Programme: HORIZON-JU-SNS-2022-STREAM-A-01-06

Start Date: 01 January 2023

Duration: 36 Months

Co-funded by the European Union

6GSNS

NANCY project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101096456.

Document Control Page

Deliverable Name	NANCY Network Information Framework
Deliverable Number	D2.3
Work Package	WP2 'Usage Scenario and B-RAN Modelling, Network Requirements, and Research Framework'
Associated Task	T2.3 'Network Information Framework Development'
Dissemination Level	Public
Due Date	30 June 2025 (M30)
Completion Date	26 June 2025
Submission Date	30 June 2025
Deliverable Lead Partner	8BELLS
Deliverable Author(s)	Ilias Theodoropoulos (8BELLS), Stratos Vamvourellis (8BELLS), Athanasios Tziouvaras (Bi2S), Ramon Sanchez-Iborra (UMU), Rodrigo Asensio-Garriga (UMU), Stylianos Trevlakis (INNO), Lamprini Mitsiou (INNO), Eirini Gkarnetidou (INNO), Giorgos-Nektarios Panayotidis (CERTH), Theofanis Xifilidis (CERTH), Dimitris Kavallieros (CERTH), Charalambos Eleftheriadis (SID), George Michoulis (SID), George Niotis (SID), Dimitrios-Christos Asimopoulos (MINDS), Ioannis Makris (MINDS), Nikolaos Ntampakis (MINDS), Panagiotis Sarigiannidis (UOWM), Thomas Lagkas (UOWM), Athanasios Liatifis (UOWM), Dimitrios Pliatsios (UOWM), Sotirios Tegos (UOWM), Nikolaos Mitsiou (UOWM), Vasiliki Koutsioumpa (UOWM), Pigi Papanikolaou (UOWM)
Version	1.0

Document History

Version	Date	Change History	Author(s)	Organisation
0.1	04 April 2025	ToC preparation	Ilias Theodoropoulos, Stratos Vamvourellis	8BELLS
0.2	20 April 2025	Section 5.2	Charalambos Eleftheriadis, George Michoulis, George Niotis, Dimitrios-Christos Asimopoulos, Ioannis Makris, Nikolaos Ntampakis	SID, MINDS
0.3	15 May 2025	Added text regarding the coverage probability prediction model	Tziouvaras Athanasios	Bi2S

	20 May 2025	Revised text regarding the coverage probability prediction model	Tziouvaras Athanasios	Bi2S
	23 May 2025	Section 5.1	Stratos Vamvourellis	8BELLS
0.4	28 May 2025	Added references, dataset descriptions and refined contributions	Tziouvaras Athanasios	Bi2S
	29 May 2025	Sections 1.2, 2.1	Ilias Theodoropoulos	8BELLS
0.5	02 June 2025	Section 6.2	Stratos Vamvourellis	8BELLS
	02 June 2025	Section 3.2.2	Ramon Sanchez-Iborra, Rodrigo Asensio-Garriga	UMU
	02 June 2025	Sections 4, 6.1.2	Giorgos-Nektarios Panayotidis, Theofanis Xifilidis, Dimitris Kavallieros	CERTH
	02 June 2025	Section 2.3	Stylianos Trevlakis, Lamprini Mitsiou, Eirini Gkarnetidou	INNO
0.6	03 June 2025	Sections 5.3, 6.1.3	Stratos Vamvourellis	8BELLS
	04 June 2025	Completed Section 1 & Executive Summary, Started Section 2.2	Ilias Theodoropoulos	8BELLS
0.7	05 June 2025	Completed Sections 2 and 7	Ilias Theodoropoulos	8BELLS
	06 June 2025	Addressed comments, reviewed contributions	Stratos Vamvourellis	8BELLS
0.8	11 June 2025	Section 6.3	Stratos Vamvourellis	8BELLS
	12 June 2025	Added citations, formatting tables, figures etc. 1st Draft Ready for Internal Review	Stratos Vamvourellis	8BELLS
0.9	20 June 2025	Addressing internal review comments	Ilias Theodoropoulos, Stratos Vamvourellis, Tziouvaras Athanasios, Giorgos-Nektarios Panayotidis	8BELLS, Bi2S, CERTH

1.0	26 June 2025	Final version after quality revisions	Panagiotis Sarigiannidis, Thomas Lagkas, Athanasios Liatifis, Dimitrios Pliatsios, Sotirios Tegos, Nikolaos Mitsiou, Vasiliki Koutsioumpa, Pigi Papanikolaou	UOWM

Internal Review History

Name	Organisation	Date
Stylianos Trevlakis	INNO	16 June 2025
Maria Belesioti	OTE	17 June 2025

Quality Manager Revision

Name	Organisation	Date
Anna Triantafyllou, Dimitrios Pliatsios	UOWM	26 June 2025

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Explanation
AINQM	Artificial Intelligence Network Quality Module
ANN	Artificial Neural Network
API	Application Programming Interface
ARIMA	AutoRegressive Integrated Moving Average
AUC	Area Under Curve
B5G	Beyond 5G
BFT	Byzantine Fault Tolerance
B-RANs	Blockchain-based Radio Access Networks
BS	Base Station
CDMA/H-DR	Code Division Multiple Access / High Data Rate
CFT	Crash Fault Tolerant
CQI	Channel Quality Indicator
dB	Decibel
dBm	Decibels relative to one milliwatt
emBB	Enhanced Mobile Broadband
FDD	Frequency Division Duplex
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDOP/PDOP	Horizontal/Position Dilution of Precision
IBFT	Istanbul Byzantine Fault Tolerance
LoS	line-of-sight
LTE	Long Term Evolution
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
Mbps	Megabits per second
MGF	Moment Generating Function
ML	Machine Learning
MLP	Multi Layered Perceptron
mmTC	Massive Machine Type Communications
MSE	Mean Squared Error
MTC	Machine Type Communications
NCGI	Network Cell Global Identity
NIF	Network Information Framework
NR5G	New Radio 5G
O-RAN	Open Radio Access Network
ORM	Object-Relational Mapping
PCI	Physical Cell ID
PDF	Probability Density Function
PLMN	Public Land Mobile Network
PPP	Poisson Point Process
QBFT	Quorum-based Byzantine Fault Tolerance
QMI WDS	Qualcomm MSM Interface Wireless Data Services
RAFT	Reliable, Replicated, and Fault Tolerant

RAT	Radio Access Technology
ROC Curve	Receiver Operating Characteristic Curve
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
SBFT	Scalable Byzantine Fault Tolerance
SINR	Signal to Interference plus Noise Ratio
SINR	Signal to Interference and Noise Ratio
SNR	Signal to Noise Ratio
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TEEs	Trusted Execution Environments
TPS	Transactions Per Second / Throughput
TX	Transactions
TX rate	Transmission Rate
UDP	User Datagram Protocol
UE	User Equipment
UI	User Interface
UMTS	Universal Mobile Telecommunications Service
URLLC	Ultra-Reliable Low Latency Communications
XGBoost	eXtreme Gradient Boosting

Executive summary

This deliverable presents the development and implementation of a comprehensive Network Information Framework (NIF) tailored specifically to help users assess and optimize Blockchain-based Radio Access Networks (B-RANs) across diverse deployment environments. Central to this initiative was the creation of three advanced AI-driven predictive models targeting key network performance indicators: coverage probability, outage probability, and network latency.

The **Coverage Probability Prediction Model** was trained on datasets reflecting realistic urban environments, ensuring its effectiveness in predicting the reliability and reach of network coverage within densely populated areas. One of the datasets used by the model was produced by NANCY at the University of Murcia's 5G deployment. Meanwhile, the **Outage Probability Prediction Model** capitalized on the established and robust urban Colosseum dataset [1], known for its precise simulation of urban network disruptions, to enhance the accuracy of predictions regarding network outages.

The **Latency Prediction Model** evaluates the performance implications of various blockchain consensus mechanisms. It was developed and validated using datasets derived from four distinct blockchain consensus protocols, providing comparative insights and decision-making capabilities in selecting consensus strategies aligned with performance goals. In order to have a more complete analysis, the trade-offs between complexity and security have also been assessed and documented.

The developed AI models have been integrated into an interactive web-based platform, enabling users to upload datasets, execute predictions, and conduct immediate network performance evaluations.

1. Introduction

B-RANs represent a novel integration of blockchain's decentralized security and control mechanisms into wireless network architectures. As wireless networks face increasing complexity and higher performance demands, robust predictive frameworks are crucial for accurate and reliable performance assessments [2]. This deliverable introduces a NIF equipped with AI-driven predictive models designed specifically to quantify coverage probability, outage probability, and blockchain-induced network latency in B-RAN environments.

1.1. Purpose of the Deliverable

This deliverable aims to document the development and validation of the NIF developed under T2.3 'Network Information Framework Development'and its corresponding AI-driven predictive models, specifically focusing on the three aforementioned key performance metrics, within B-RAN configurations. The training of these models is demonstrated using real-world datasets that are representative of typical urban environments and diverse blockchain consensus mechanisms. The mathematical foundations and machine-learning workflows are detailed, and the integration of these models into a user-friendly web platform is outlined. Through this platform, custom network parameters can be uploaded, consensus configurations can be selected, and instant performance assessments can be obtained by network engineers and researchers, allowing for data-driven planning and optimization. Ultimately, both a theoretical exposition of the modelling approach and practical guidance for employing the NIF to evaluate and enhance B-RAN deployments are provided.

1.2. Structure of the Document

This deliverable is structured in a systematic manner to provide a thorough and coherent overview of the work conducted throughout T2.3 'Network Information Framework Development'. The document consists of seven main sections, each focusing on a specific area of the task and collectively offering a complete perspective on the research, methodologies, evaluations, and outcomes.

The deliverable structure is as follows:

- **Section 2 - NIF Overiew** provides the fundamental aspects of the NIF. It explores the architecture of the framework, its placement within the NANCY platform, and the functionalities of the graphical user interface (GUI). This section establishes the technical foundation for understanding how the NIF integrates into the broader ecosystem and the significance of its design.
- **Section 3 – Coverage Probability Prediction** documents the development of the coverage probability prediction model, along with the data collection and training approaches.
- **Section 4 – Outage Probability Predection** discusses the coverage outage prediction model, as well as the data collection and training approaches.
- **Section 5- Latency Prediction** describes the model used to predict the blockchain-induced latency, the data collection method, and the training approach.
- **Section 6- Performance Evaluation** is devoted to evaluating the aforementioned models. It includes a comprehensive evaluation of the performance of individual models, which is conducted using both quantitative data and mathematical foundations such as point processes. The section also examines the trade-off between security and complexity in consensus mechanisms.
- **Section – 7 Conclusion** provides a synthesis of the key findings and concludes the deliverable.

2. NIF Overview

2.1. NIF Architecture

2.1.1. Architectural components

NIF's architecture (Figure 1) is designed to be highly modular, interactive, and user-centric, ensuring that each component contributes meaningfully to a seamless user experience and robust analytical capability. The architecture integrates several interdependent modules, each having a specific role that supports secure access, efficient data processing, predictive modelling, and insightful visualizations.

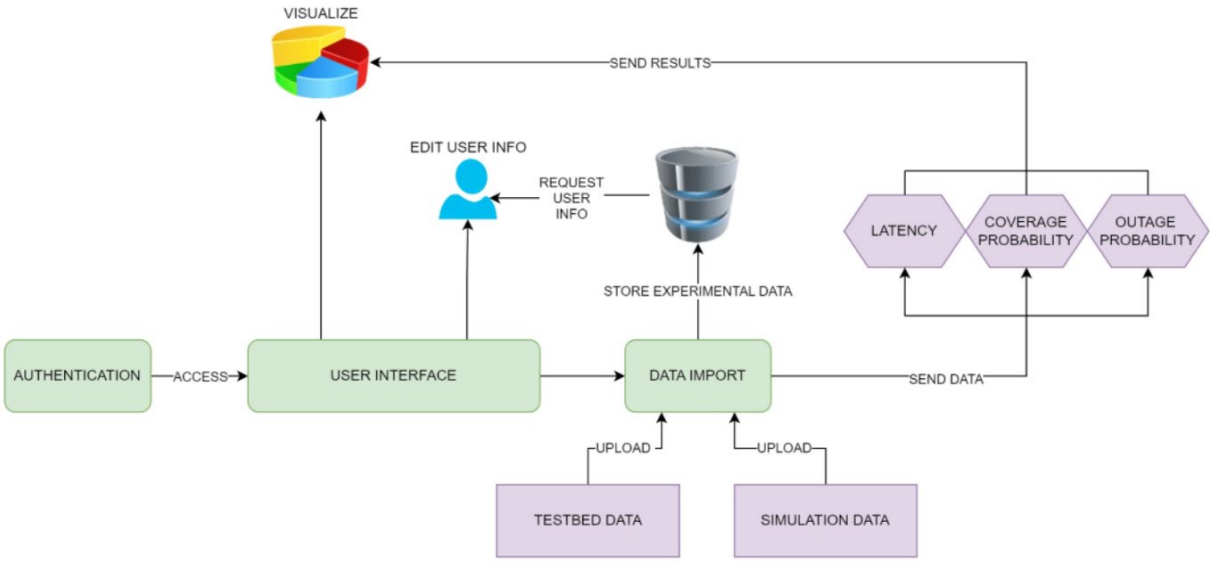

Figure 1: NIF's Architecture

At the foundation of the system lies the **Authentication Module**, which governs secure access to the platform. This module ensures that only verified users interact with the system's functionalities. It acts like the first point of entry, employing protocols such as credentials for safeguarding data and preventing unauthorized access. This is particularly crucial in operational settings where data integrity must be preserved.

Once authenticated, the user interacts primarily through the **User Interface** (UI). This interface is designed for accessibility and efficiency, enabling both novice and expert users to navigate the system with ease. Through the UI, they can perform many actions: upload datasets from testbeds, configure the scope of analysis, and see the output predictions of the models generated. UI plays a critical role in simplifying analytical workflows, offering intuitive controls and guidance that help users at each step.

Data ingestion is being handled by the **Data Import Module**, which manages user-uploaded datasets. This module supports a variety of data formats, including CSVs and JSONs, making it compatible with experimental and simulation datasets.

After successful uploading, the data is stored persistently in the **Database**, a central repository underpinning the whole framework. The database ensures all imported data, along with metadata and configurations, is stored in a structured manner. The database also serves as a source for feeding the data into the predictive engines.

At the NIF's core lie the **Predictive Models**, a group of AI models developed to quantify network metrics. These include:

- The Blockchain Latency Prediction Model, which estimates delays in the network under multiple consensus mechanisms.
- The Coverage Probability Prediction Model, which estimates network signal coverage in urban areas.
- The Outage Probability Prediction Model, which calculates the service interruptions probability based on real-world data.

These predictive models' outputs are routed into the **Visualization Module,** which translates analytics into visual formats. It renders dashboards and charts that show latency, coverage or outage likelihoods. These visuals make results more accessible and allow users to make informed decisions. One can explore the effects of input parameters, compare datasets or derive insights directly from the UI.

2.1.2. Technical Implementation Details

Framework Selection

NIF is built with Django (Python). Database access is handled entirely through Django's Object-Relational Mapping (ORM) with PostgreSQL, supporting migrations and avoiding raw SQL. The authentication system is extended with role-based permissions for different user groups. The Django admin interface is customized to manage user accounts and dataset metadata. The AI models are invoked through Django views and the visualization images are generated server-side and delivered to the client as static files; users interact with the system via HTML forms and JavaScript (Fetch API) without a front-end framework [3].

Authentication Protocols

User authentication in NIF is implemented using Django's built-in authentication system, which provides a secure and extensible foundation. During registration, users provide a username, first name, last name, email address, organization affiliation, and password. All submitted data is processed through validated Django forms to ensure correctness and security. The login and logout mechanisms rely on Django's session-based authentication, maintaining user state through secure server-side sessions, with CSRF protection automatically applied to all forms and views to prevent cross-site request forgery attacks. Password management includes a reset process that generates unique email-based tokens, allowing users to securely reset their passwords when necessary. The system is designed to support role-based access control through Django's Groups and Permissions framework, which is configured to enable future differentiation of user roles, such as regular users, contributors, and administrators. All communications between the client and server are transmitted over HTTPS, ensuring encrypted and secure data exchange across the entire platform.

Django's authentication system follows strong security practices. Passwords are hashed using the PBKDF2 algorithm by default, with the option to use stronger ones like Argon2 to better protect against brute-force attacks. The framework also includes built-in protections against common attacks such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). Django regularly updates its authentication features to fix new security issues, helping keep applications safe and up to date [3].

Data Flow Mechanisms

Users begin by uploading CSV or JSON data files through an HTML form, which are saved to the server's file system under a structured media directory. Metadata about the uploaded files is stored in the database.

From the visualization dashboard, the user selects a file and requests a specific analysis (latency, outage, coverage). The client sends an HTTP GET request to a Django view with the selected file and analysis type.

The Django view loads the requested CSV/JSON file and performs the model inference using the aforementioned models. The model outputs are formatted and passed to Matplotlib to generate one or more visualizations. The resulting plots are saved as static PNG images on the server.

Finally, the browser receives the filename of the generated image from the server and loads the image into the page. All communication between the client and server is performed through Fetch API calls.

Code Structure

NIF follows Django's standard project structure, organized into reusable applications (apps):

- **accounts**: This is a standard Django app for user management, built on Django's authentication framework. It handles login, logout, registration, session management, and role-based permissions. It also manages file uploads, file metadata and extends the Django admin interface for managing users and datasets.

- **models_app**: This is a custom Django app that integrates the AI models developed specifically for NIF. It provides views and templates for running these models on uploaded datasets and generating visual output. The app handles the execution of model inference and produces server-side visualizations that are displayed to the user through the web interface.

Database Schema

The NIF database uses PostgreSQL with the following schema:

- **Users:** username, first name, last name, organization, email, hashed password, account creation and last login timestamps.

- **Datasets:** filename, upload date, file type, file size, owner (user ID)

2.1.3. NIF Hosting

The 8BELLS NIF hosting infrastructure provides a robust and reliable environment for managing user interactions and dataset storage. It features a secure backend database that supports the storage and retrieval of user information and experimental data efficiently. Additionally, the platform is equipped with sufficient computational resources to handle the AI/ML model workloads for fast inference.

2.2. NIF Graphical User Interface

The NIF features an intuitive and user-centric GUI designed to streamline interaction and enhance the efficiency of data-driven analytics. The interface promotes ease of use while supporting complex predictive functionalities in an accessible manner.

2.2.1. Authentication

The authentication module establishes the first layer of user interaction, delivering a secure and intuitive process for accessing the framework. Upon accessing the application, users are greeted with distinct Sign Up (Figure 3) and Log In (Figure 2) portals, each crafted with clarity and ease-of-use in mind. Sign-up requires entering essential personal information-Username, First Name, Last Name, Email, Organization and a secure Password-ensuring proper user identification and future communication capabilities.

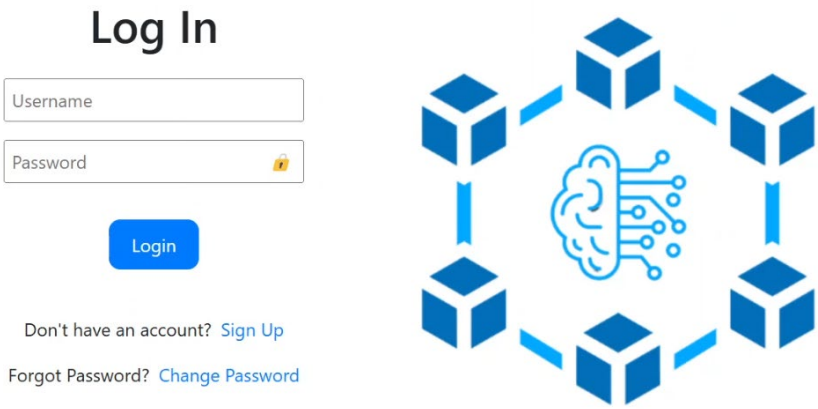

Figure 2: Authentication Module

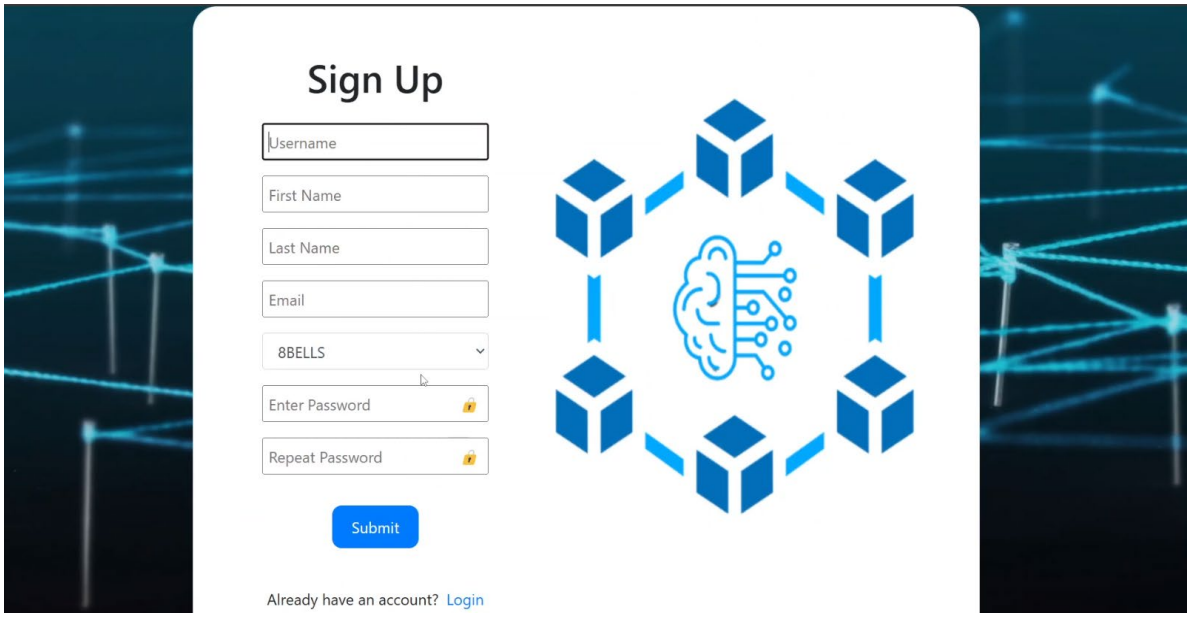

Figure 3: Sign Up Page

To enhance the onboarding experience, the sign-up interface integrates modern UX features such as real-time field validation, tooltips for password requirements, and visual indicators for input errors. Organization selection is facilitated through a searchable dropdown, promoting accurate user affiliation with minimal effort. Upon successful account creation, a confirmation message (Figure 4) reassures users and guides them towards logging in. The Log In screen supports password visibility toggling and a “Forgot Password” workflow (Figure 5), ensuring accessibility in case of credential issues.

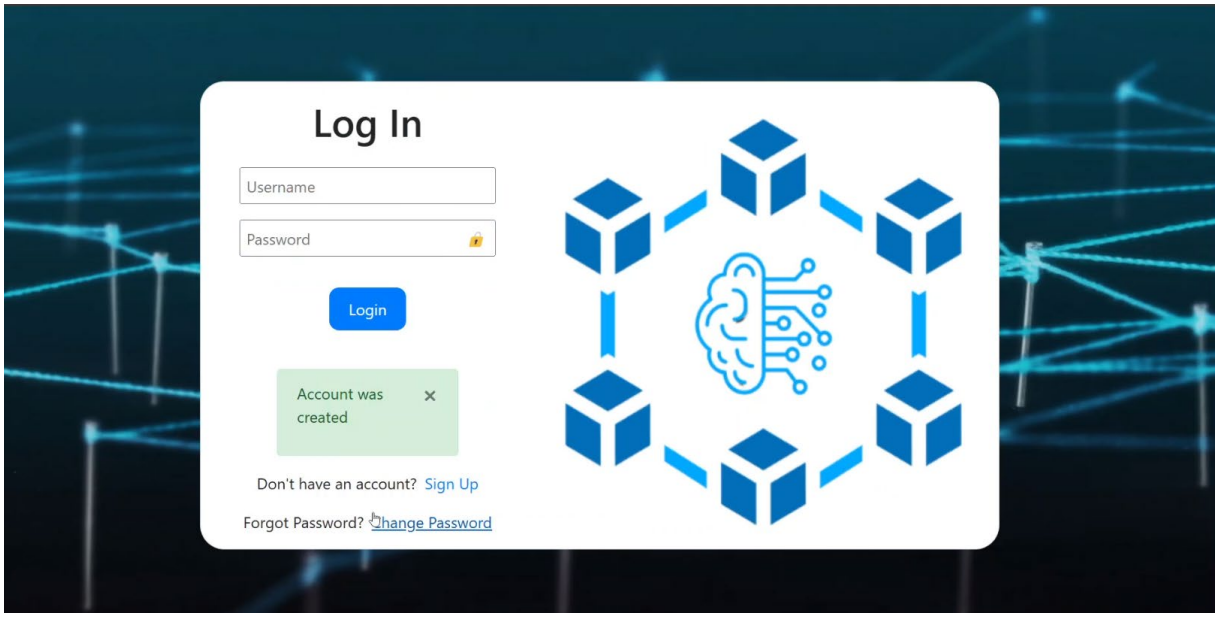

Figure 4: Success Message

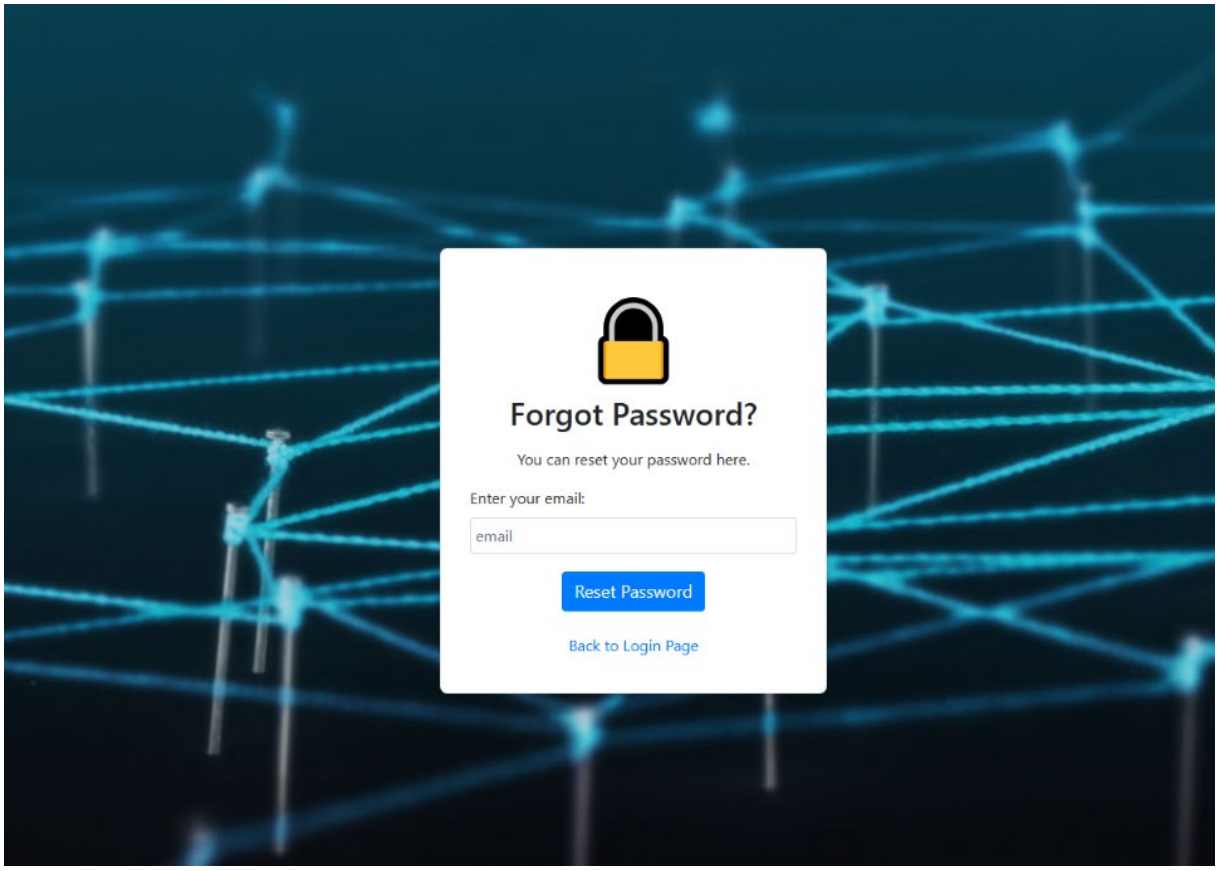

Figure 5: Forgot Password Workflow

2.2.2. Home Dashboard

The Home Dashboard (Figure 6) serves as the central hub of the application, immediately presenting users with the core functionalities offered by the NIF. Designed for quick access and high-level visibility, the dashboard provides entry points to three principal analytical modules: Latency Prediction, Coverage Prediction, and Outage Prediction. Each option is visually distinct and accompanied by descriptive text to help users understand its purpose.

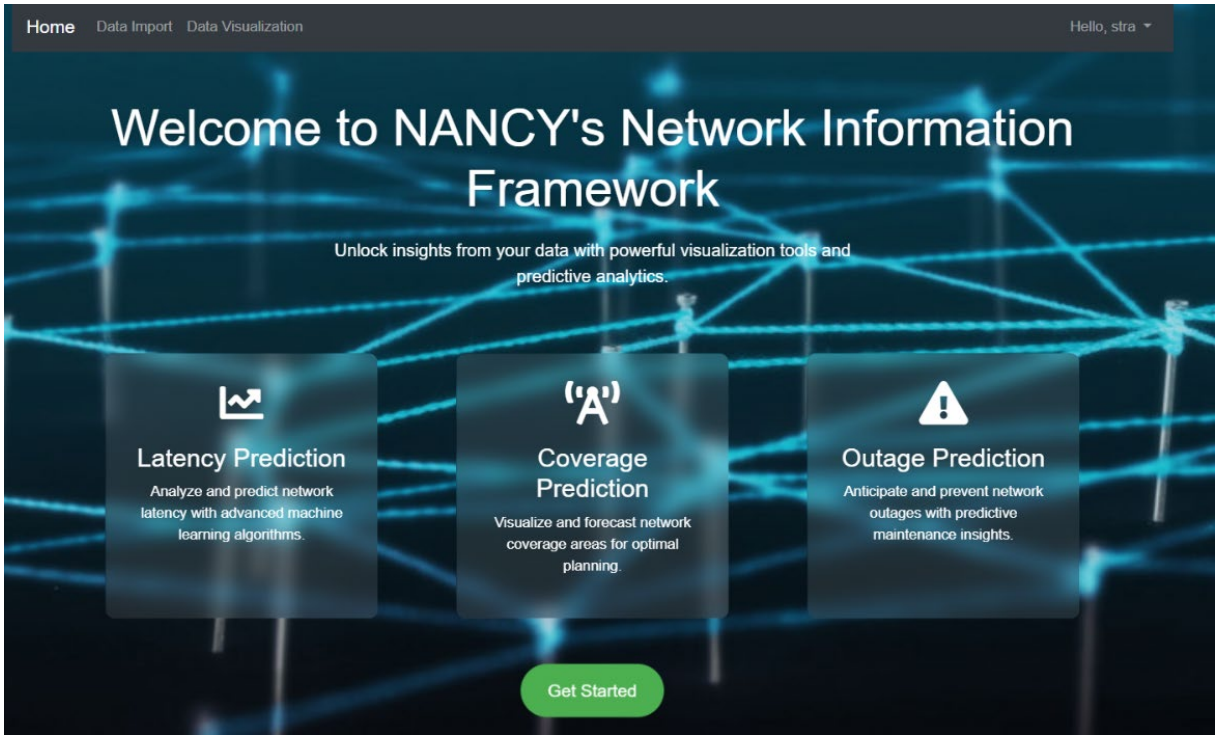

Figure 6: Home Dashboard

2.2.3. Data Import

The Data Import module (Figure 7) is a vital component of the framework, enabling integration of external datasets. Users can upload datasets via a clean, drag-and-drop interface or through a traditional file selector. Accepted file formats (such as CSVs) are clearly indicated, and users are prompted to name their files for easy identification. Real-time progress indicators, validation messages (Figure 8), and post-upload confirmations contribute to a smooth user experience.

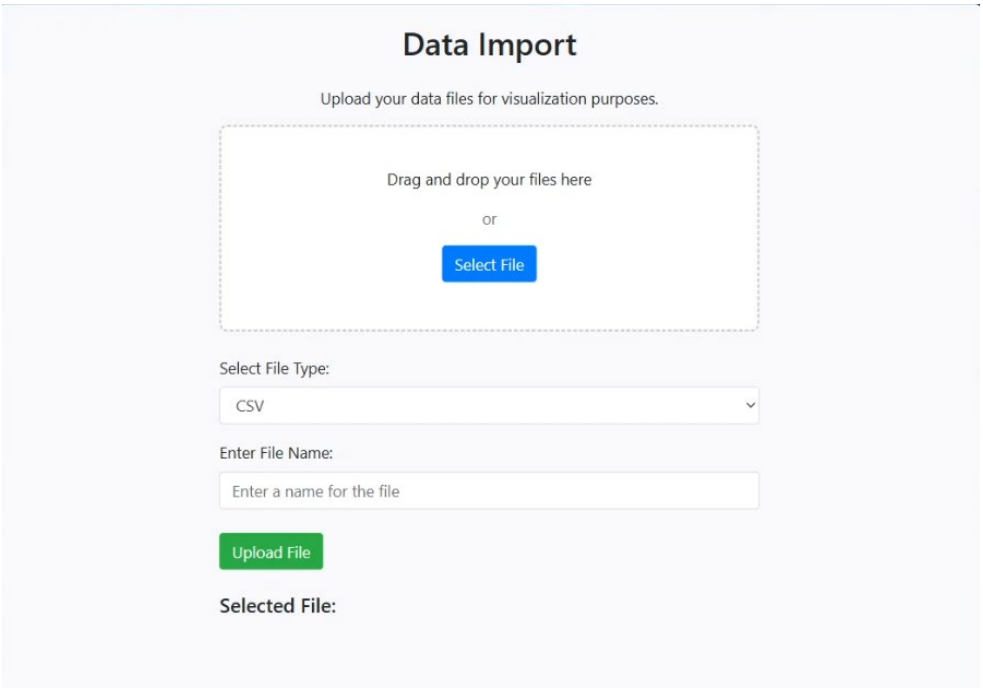

Figure 7: Data Import Module

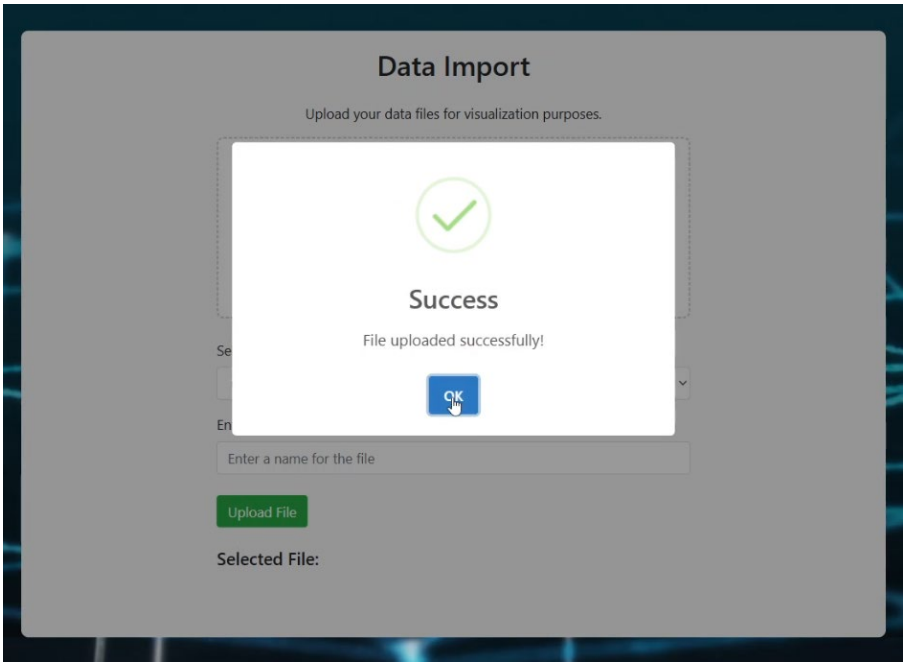

Figure 8: Successful Upload Message

2.2.4. File Management

The File Management interface (Figure 9) provides a comprehensive and organized view of all user-uploaded datasets. It includes a table that lists each dataset along with key metadata such as file name, upload date, uploader identity, file size, and data type. Users can take action on each dataset, including viewing its contents, deleting it, or preparing it for analysis.

To enhance user control and transparency, the interface supports bulk actions and confirmation prompts before deletions. A file versioning system may also be included to track changes over time, allowing users to revert or compare datasets. This module is crucial for maintaining a clean and efficient data workspace, especially when handling multiple projects or collaborative workflows across teams.

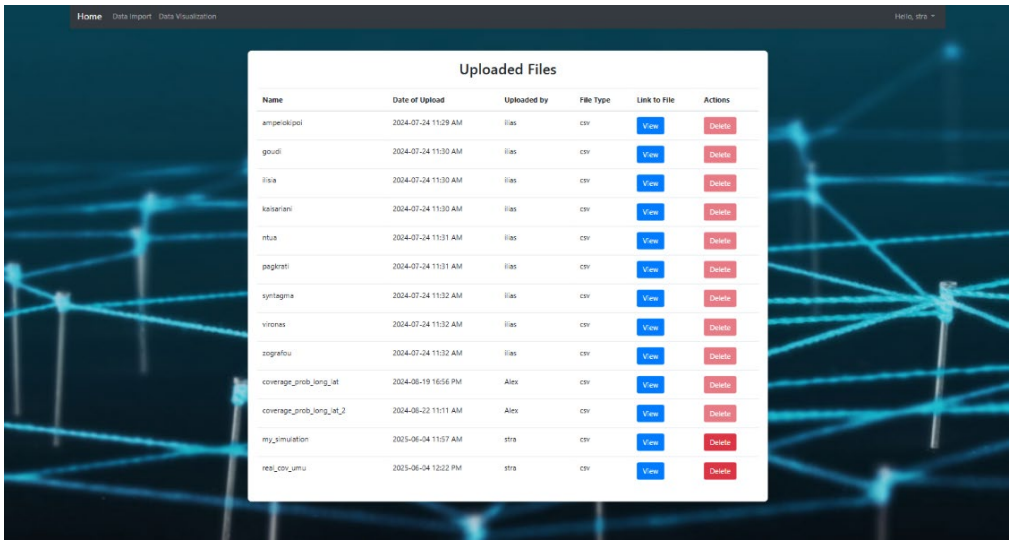

Figure 9: File Management Page

2.2.5. Visualization Dashboard

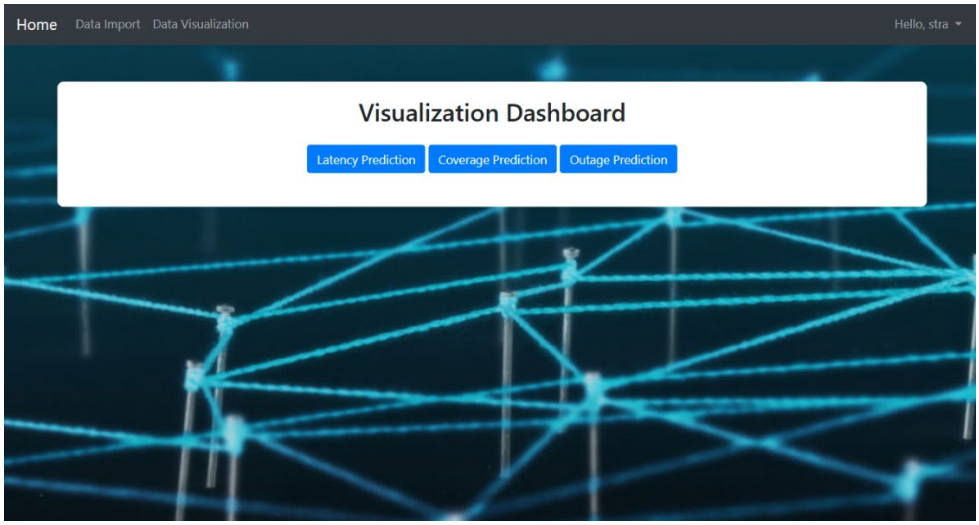

Figure 10: Visualization Dashboard

The Visualization Dashboard (Figure 10) is the most critical component of the NIF, as it serves as the starting point for prediction processes and generates the corresponding visual outputs. Users can choose a specific prediction engine and apply it to a selected dataset to initiate analysis. Prediction results are rendered in real time on the same page, enabling users to view and interpret outcomes without navigating away from the dashboard.

Outage prediction

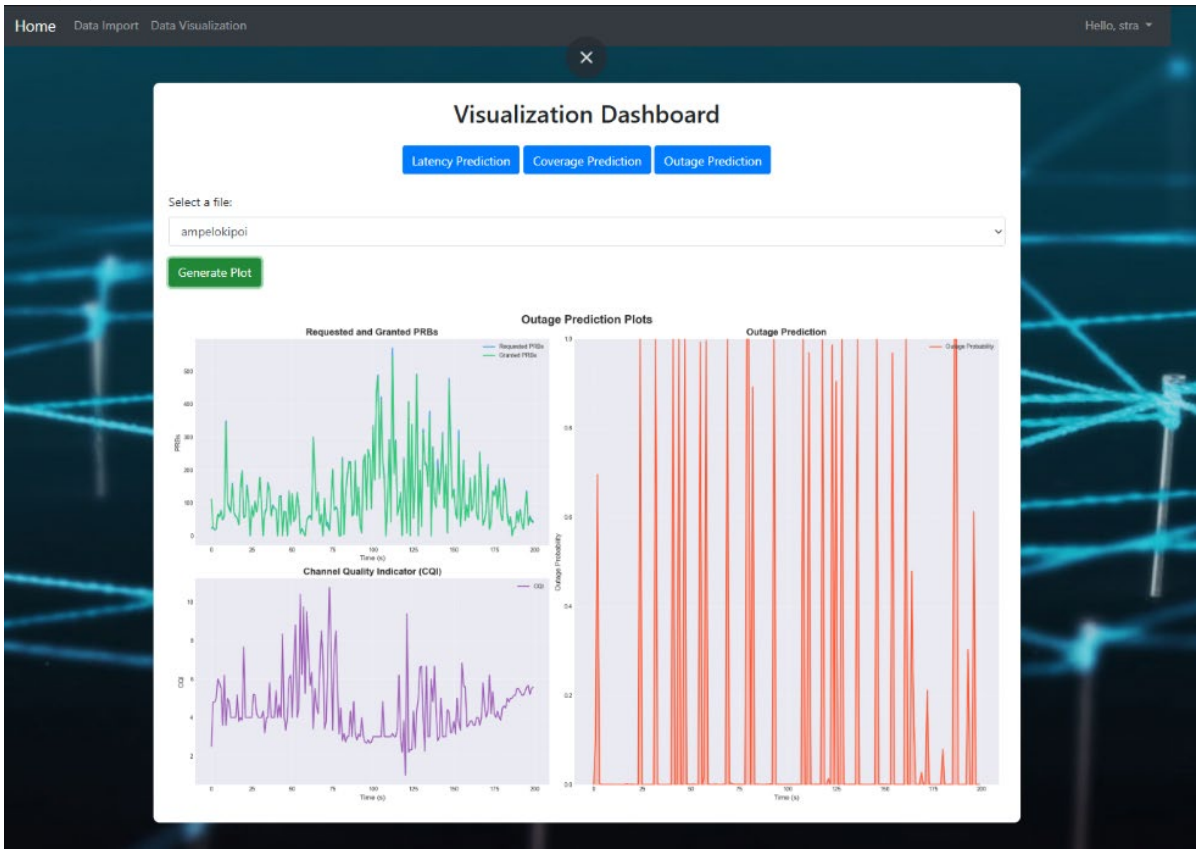

Figure 11: Outage Prediction

Upon generating the plot (Figure 11), the dashboard displays three time-series visualizations. The two plots on the left represent two of the input features used by the outage prediction model, while the plot on the right displays the model's output, indicating predicted outage events over time.

The top-left plot shows the Requested and Granted PRBs over time. PRBs, or Physical Resource Blocks, are fundamental units of resource allocation in 5G networks. This graph tracks how many PRBs are requested versus how many are granted at each time point. The difference between these two metrics can be indicative of network congestion or resource scarcity conditions that may contribute to service outages. A significant gap between requested and granted PRBs suggests that user or system demands are not being fully met, which could degrade service quality.

Beneath it, the bottom-left plot illustrates the "Channel Quality Indicator (CQI)" as it changes over time. The CQI provides a measure of the communication channel's quality, which reflects signal strength, interference, and overall link reliability. Lower CQI values denote poorer channel conditions, which are known precursors to data transmission failures and outages. Monitoring CQI allows the system to assess environmental and network-level conditions that affect connectivity performance.

The right-hand plot displays the output of the outage prediction model and shows the model's prediction for whether an outage is likely to occur at each point in time. The Y-axis ranges from 0 to 1, where a value of 1 signifies a predicted outage, and 0 signifies no outage. This binary prediction pattern allows network operators to visualize the temporal distribution of expected service failures. Frequent spikes to 1 suggest multiple predicted outage events across the observed timeframe.

Coverage prediction

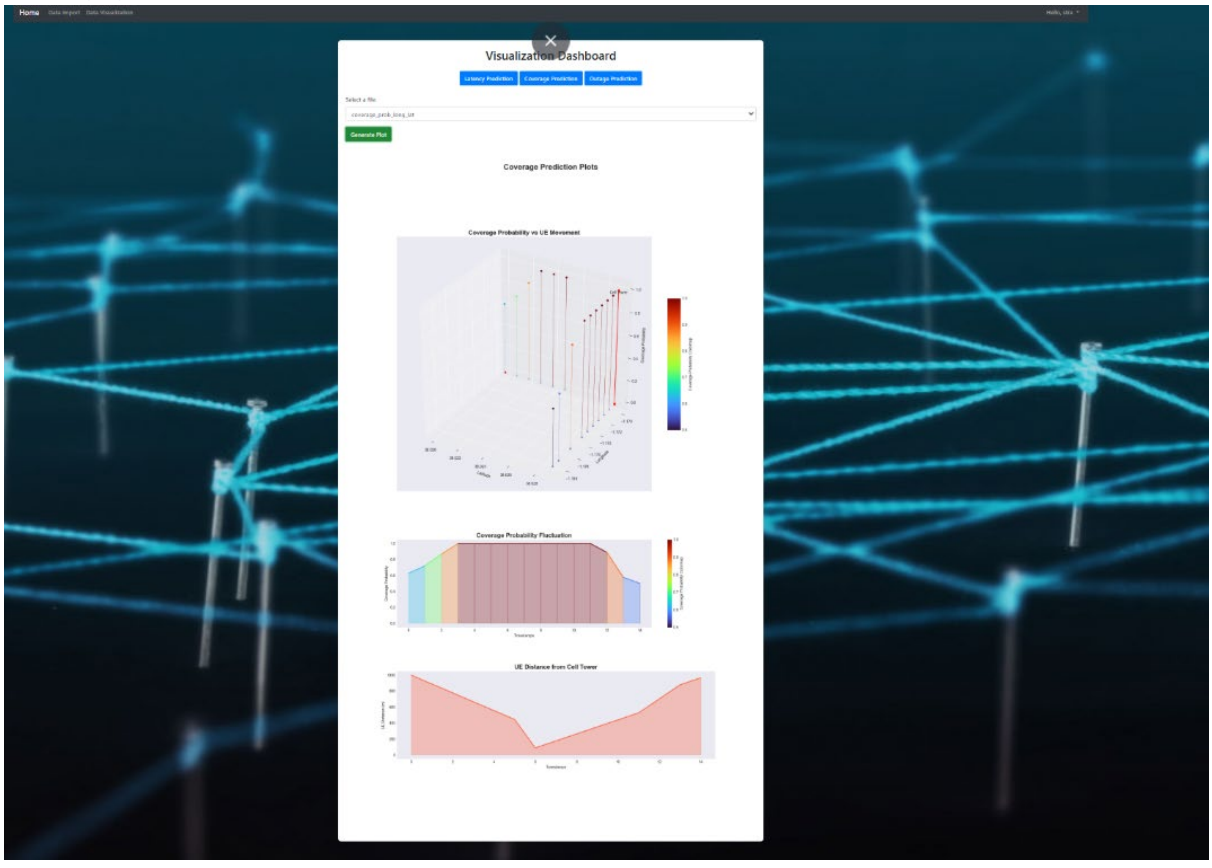

Figure 12: Coverage Prediction (1)

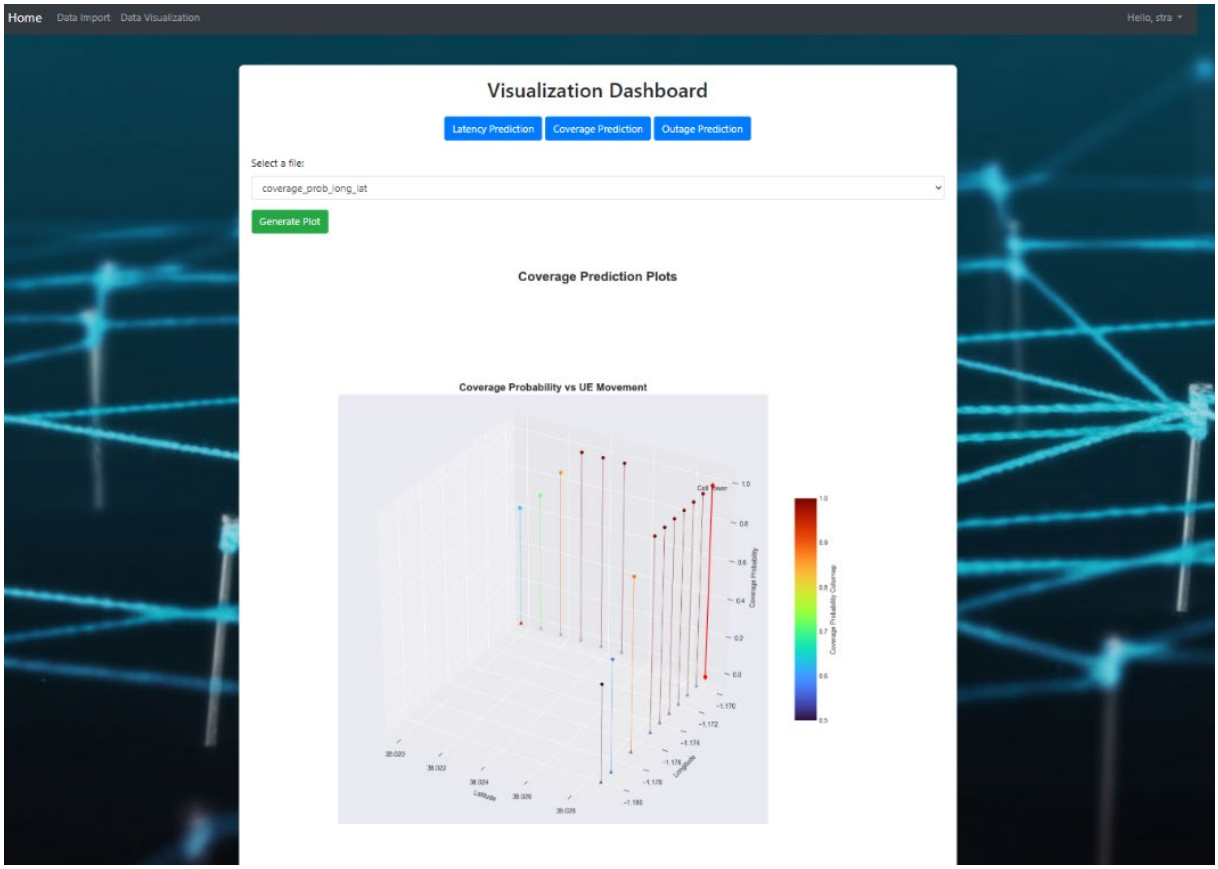

Figure 13: Coverage Prediction (2)

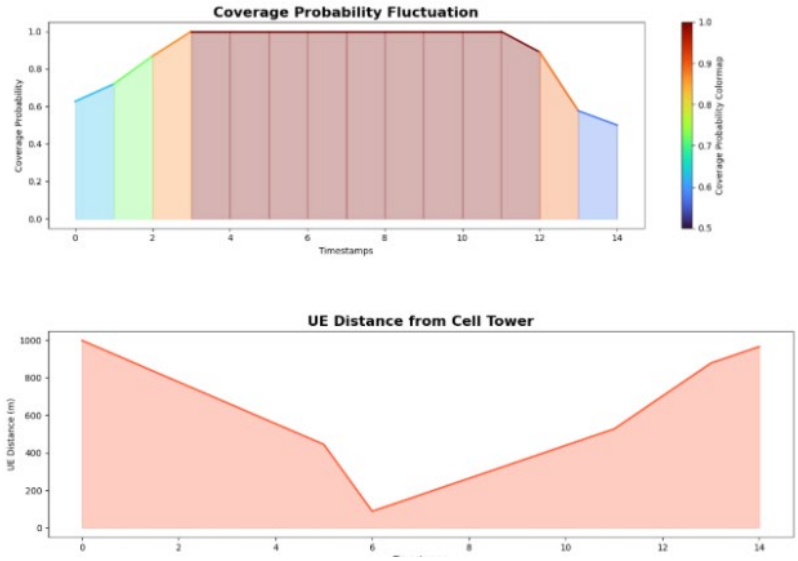

Figure 14: Coverage Prediction (3)

The visual output consists of three distinct but interrelated plots, as shown in Figure 12.

The first visualization (Figure 13) is a **3D plot** depicting coverage probability in relation to **user movement**. The axes represent:

- **X-axis**: Longitude

- **Y-axis**: Latitude
- **Z-axis (Color scale)**: Coverage probability

The color gradient, ranging from blue (low) to red (high), indicates the **likelihood of maintaining coverage** at each spatial point. Areas with deep blue shades signal low coverage probability, potentially corresponding to physical obstructions, distance from the Base Station (BS), or environmental interference. This spatial mapping is crucial for identifying **coverage blind spots** or **weak signal regions**. The cell tower is also marked in this plot.

The second plot (Figure 14) is a **time series visualization** showing the fluctuation of **coverage probability** over a sequence of measurement intervals. The Y-axis reflects the estimated coverage probability, while the X-axis represents discrete time steps or events. This chart emphasizes how coverage conditions vary dynamically as the User Equipment (UE) moves. Sharp declines or instability in this graph can be early indicators of potential handovers, interference zones, or network edge transitions.

The final plot visualizes the **distance of the UE from the serving cell tower** over time. This graph provides essential context for the other plots, as distance is a primary factor influencing signal quality and coverage reliability. The Y-axis shows the physical distance (likely in meters), and the X-axis indicates time or event count.

Latency prediction

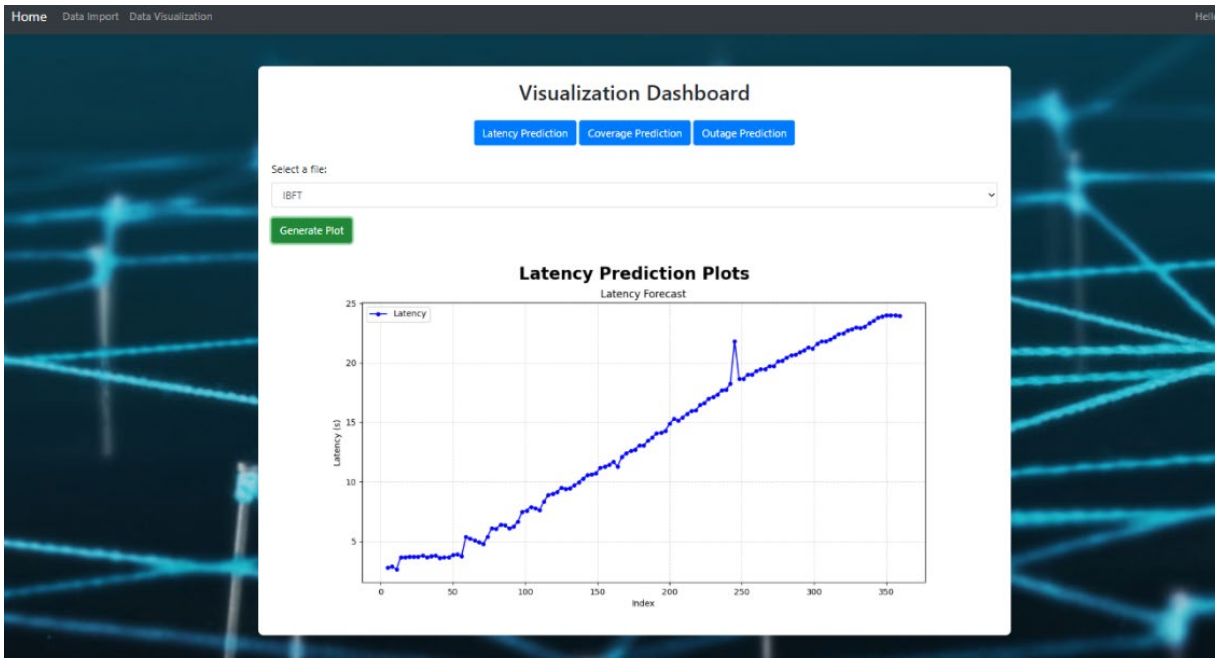

Figure 15: Latency Prediction

The main component of the output is a single time-series graph (Figure 15), which captures the model's estimate of latency values across a range of time-indexed events. This particular plot exhibits a clear upward trajectory, suggesting a gradual increase in latency as the sequence progresses. This rising pattern may imply growing congestion, potentially due to system overload. The number of orderers of each consensus mechanism also plays a pivotal role in the system's latency, as increasing it corresponds to increased architectural complexity.

2.2.6. User Information

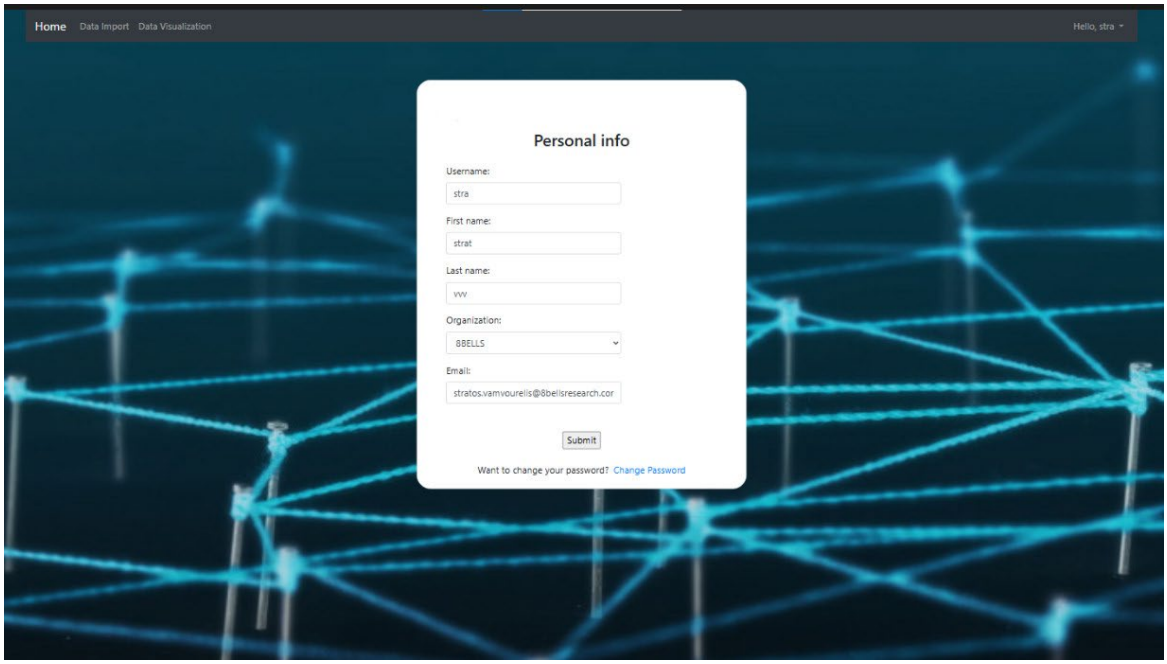

Figure 16: User Information

The User Information Management page (Figure 16) offers users complete control over their account details. Accessible via a dedicated My Account tab, this section displays fields such as Username, First Name, Last Name, Organization, and Email in an organized layout. Users can review their data and update their information at any time.

2.3. NIF Placement within NANCY

This section aims to provide a clear placement of the NIF component within the functional and deployment view of the NANCY architecture. The NIF is designed to be a highly modular, interactive, and user-centric framework. It integrates several interdependent modules, each contributing meaningfully to a seamless user experience and robust analytical capability. The architecture of the NIF, depicted in Figure 1, integrates modules that support secure access, efficient data processing, predictive modelling, and insightful visualizations. As illustrated in Figure 17, these modules function together to provide a full toolset aimed at helping users optimize B-RAN system's performance.

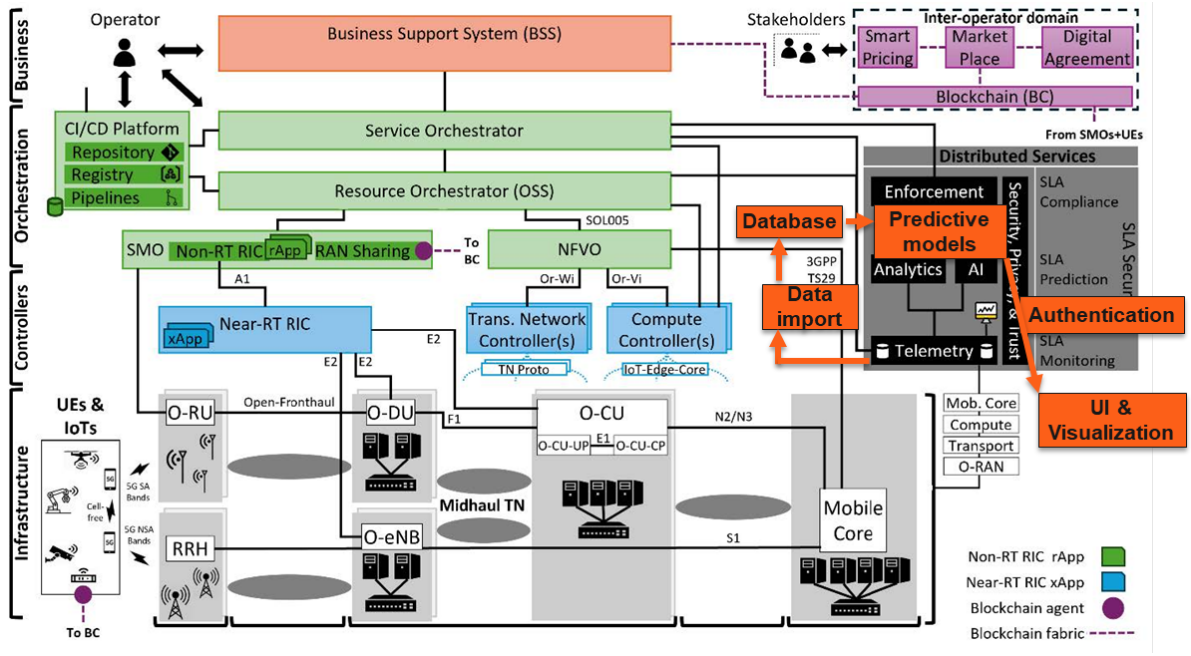

Figure 17: Functional and deployment view of the NANCY architecture

In the NANCY project, the NIF provides the necessary AI-aided analytical capabilities for understanding and improving network performance in Beyond 5G (B5G) environments. Its ability to predict key performance indicators like latency, coverage, and outage probability allows users to gain nuanced insights and optimize B-RAN systems.

3. Coverage Probability Prediction

This component leverages an AI model to predict the Reference Signal Received Quality (RSRQ) of the UE. RSRQ, in turn, can be used to estimate the coverage probability of the UE. Thus, our effort focuses on designing, training and fine-tuning an AI model, capable of producing robust and accurate predictions which can be used by NANCY's NIF to assess the coverage probability.

In our case, the term AI is used to describe a collection of algorithms and techniques that produce a model capable of making predictions, given some input data. In general, the process of designing and implementing AI models can be compartmentalised into the following phases: **(i)** model design; **(ii)** model training; **(iii)** model testing, also known as inference. During the model design phase, the main goal is to find the most appropriate AI model architecture that can fit the data under investigation. For this reason, we have performed an assessment of existing state-of-the-art methods, while also considering the characteristics of the data which will be used for model training. During the model training operation, the model is fed with a training dataset, which is a collection of data samples, and tries to fine-tune its parameters (also known as weights) in order to better represent the input data. When the AI model's weights are adjusted to properly represent the input data, the training process is completed. In the sequel, during the model testing procedure, the trained AI model is tested with data that stem outside of the original training dataset. This process reveals how efficiently the trained model generalises to other data samples. Additionally, within this operation, the trained model generates predictions using the new input data. In the following subsections, we present the AI model designed, trained and tested within the NANCY project, to predict the coverage probability of the UE.

3.1. Model Design and Methodology

For the aforementioned task, we opt to utilise a multi layered perceptron (MLP) model. MLPs consist of several layers (commonly named as "input layer", "hidden layers" and "output layers"), each layer containing a set of neurons. Each neuron uses nonlinear activation functions, allowing the network to learn complex patterns in data. Figure 18 [3], depicts the general architecture of an MLP. We chose the MLP architecture to solve the RSRQ prediction problem since it provides the following advantages:

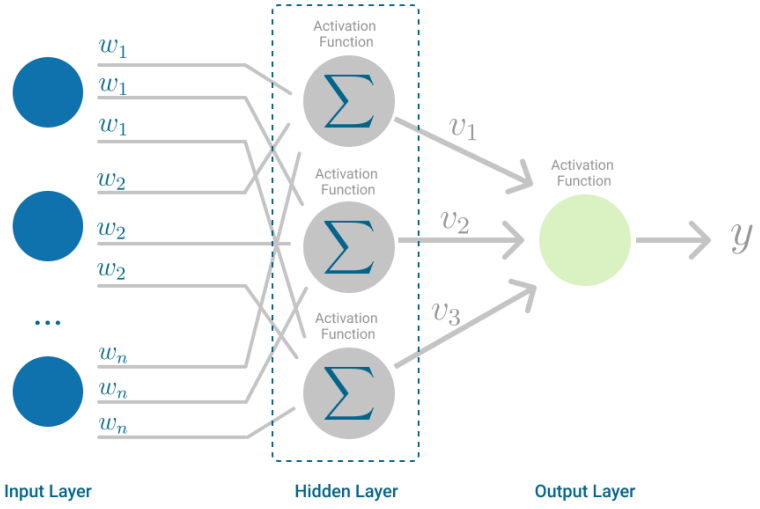

Figure 18: The overall architecture of an MLP model

1. **Lightweight and fast execution:** MLPs provide an advantage compared to other AI models since their complexity is low. This allows the model to be used in a wide range of hardware

devices (from Raspberry Pi devices to commercial processors) without requiring specialised and costly hardware (e.g. Graphics Processing Units). Our decision to design a lightweight model does not affect only the computational complexity, but also the amount of energy consumed by the model. In both training and inference operations, lightweight models consume less energy compared to complex ones. This is very important for NANCY, as the project targets green ICT goals and aims to reduce energy consumption in several key modules of its architecture.

2. **Generalisation with varying data sizes:** The generalisation property of an Artificial Neural Network (ANN) is a critical one. Generalisation measures the performance of the trained model with real-world data during the inference operation. MLP designs can reach adequate levels of generalisation, if trained properly. The advantage of MLPs is that they can reach this stage, even when trained with lower amount of data. In contrast with other architectures, which require very large training datasets, MLPs (if designed correctly) can function well with smaller datasets. This property has a direct impact on the model overfitting phenomenon, which can be catastrophic in cases where a low volume of data is fed to an AI model.
3. **Flexible fine-tuning:** Smaller networks also have higher flexibility when fine-tuning is considered. During the fine-tuning operation, which is conducted after the model training completes, a set of heuristic methods is deployed to nudge the model towards better predictions. Such methods may include layer clipping, low-rank adaptation techniques, transfer learning operations, layer freezing and model retraining. These processes often require to partially retrain the model, thus consuming energy and time. When large and complex models are under consideration, the computational requirements of the fine-tuning are large and, in most cases, forbid a detailed parameter exploration. On the other hand, smaller models can be retrained many times without several time or energy penalties, which empower designers to conduct a detailed exploration of the parameters that affect the model performance.

The methodology we employed to design this module the fine-tune the model is illustrated in Figure 19 below, and it consists of a training and an inference process.

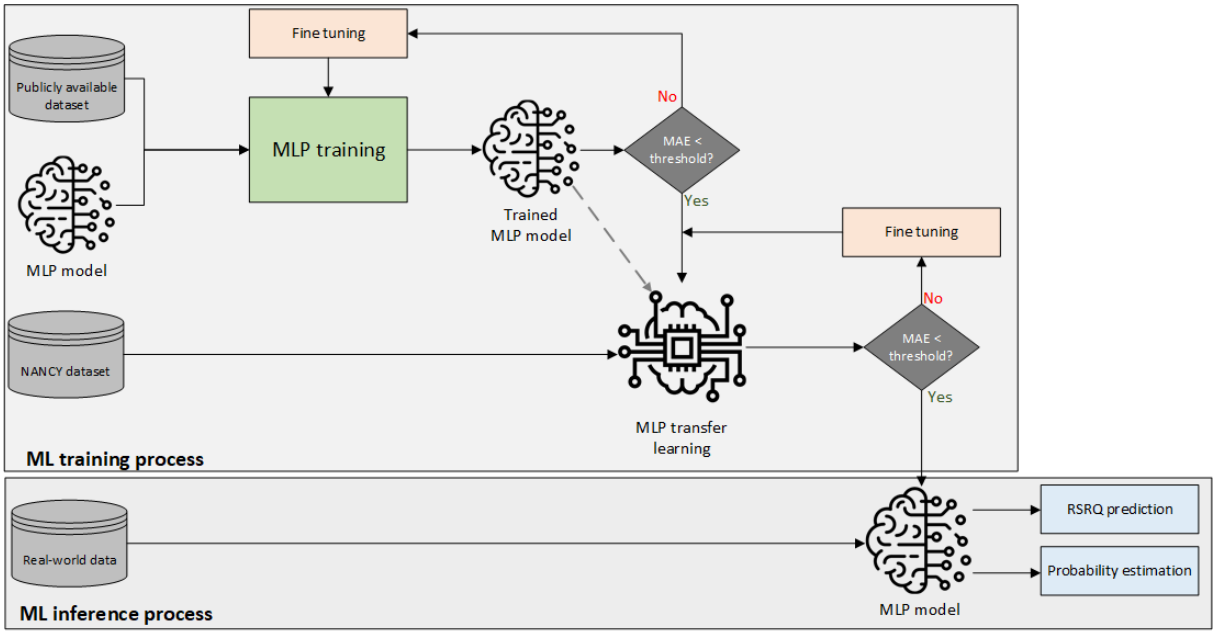

Figure 19. The methodology employed by NANCY to design and train the MLP model

During the machine learning (ML) training process, we first select publicly available data to train the MLP. More information about the training datasets that are used within this process can be found in the next subsection. The MLP model architecture along with the parameters of the training process are described Table 1. The number of layers and neurons of the model were chosen after conducting a detailed design space exploration with the training datasets. To this end, different model architectures were tested and their training performances were compared in order to deduce the optimal design that fits the training data. The current design achieved the best performance and was chosen for this reason. We should also note that more complex designs (e.g. with more layers or more neurons per layer) were prone to overfitting and thus were rejected by the consortium. On the other hand, less complex designs exhibited signs of underfitting and failed to achieve the required model quality.

Table 1. Parameter details of the MLP architecture and training process

Model type	**MLP**
Number of layers	4
Neurons for each layer	1st layer: 70 2nd layer: 70 3rd layer: 70 4th layer: 1
Activation functions	1st layer: Relu 2nd layer: Relu 3rd layer: Relu 4th layer: Linear
Training epochs	500
Optimizer	Adam
Loss functions	Mean Absolute Error (MAE)

Batch size	10

The activation function used in 1st, 2nd and 3rd layers is the Relu, which can be described using the following formula:

$$f(x) = \begin{cases} x, & if\ x > 0 \\ 0, & if\ x \leq 0 \end{cases}$$

where x represents the input values of the activation functions (which are also the layer neuron outputs). On the other hand, the 4th layer of the model utilises the linear action function which can be described using the following equation:

$$f(x) = \mathrm{x}$$

The model training operation outputs a trained MLP model. In the sequel, we check if the MAE constraints are satisfied. MAE is a metric that measures the absolute difference between the predicted RSRQ value and the real one and can be calculated using the following formula:

$$MAE = \frac{\sum_1^n |y_i - x_i|}{\mathrm{n}}$$

Where y_i is the actual (real) value for data point *i*, x_i is the model output (prediction) and *n* is the total number of samples used for testing the model.

After the first round of training finishes, we commence a transfer learning operation using the dataset collected through NANCY's testbeds. More information about these data is documented in the "data collection" subsection below. Generally, transfer learning exploits the knowledge gained from a previous task to improve the model's generalisation for a new task. Transfer learning is frequently used to retrain a model using new data, instead of initiating a training process from the beginning. In our case, we leverage transfer learning to nudge the MLP's weights to better fit the data, which is collected by NANCY demonstrators. We opt to use the transfer learning technique to increase the generalisability of the model to NANCY's demonstrators. A different approach would be to merge the two training datasets (the LTE Dataset and the NANCY dataset as they are described in section 3.2 below) into one and then train the model using this merged training set. The issue with this approach is that the LTE dataset is significantly larger than the NANCY dataset. This would nullify the impact of the data distribution of NANCY's dataset on the model's training process. In this sense, the data distribution of the merged training dataset would represent the data distribution of the LTE Dataset for the purposes of model training. To account for this, we first train the model only using the LTE Dataset and then retrain it (using the transfer learning operation) with a different data distribution, i.e., the NANCY dataset. This choice not only helps the model to generalise to NANCY's demonstrators, but also highlights that knowledge transfer operations are fully supported in NANCY's AI framework. At the end of this process, we apply fine-tuning optimisations to increase the model quality.

While transfer learning produces a fully trained model which can be used for inference operations; inference utilises real-world data to perform predictions, using the trained model. The MLPs designed under this task support two types of outputs: (i) RSRQ prediction and (ii) Probability estimation. RSRQ prediction is an estimation of the RSRQ value at a given time. Probability estimation generates a range of possible RSRQ values, along with their corresponding probabilities. Both of those outputs can be used to assess the coverage probability of a UE, depending on the specifics of the application scenario.

3.2. Data Collection

Our MLP methodology utilises two distinct datasets: A dataset that consists of publicly available data and a NANCY dataset that contains data collected from the UMU testbed.

3.2.1. Publicly available data

We use the "LTE Dataset" [4] which can be downloaded from Kaggle [5]. It contains 135 traces, with an average duration of fifteen minutes per trace, with viewable throughput ranging from 0 to 173 Mbit/s at a granularity of one sample per second. The dataset consists of several measurements (such as distance from the Cell tower, UE speed, UE position, RSRP, RSRQ and SNR values) in different scenarios and within different mobility schemes (bus, car, pedestrian, static and train).

3.2.2. NANCY dataset

This dataset gathers information collected from two sampling campaigns conducted on two different days in the University of Murcia's 5G deployment located at the Espinardo Campus of this institution. It is made publicly available by UMU in Zenodo [6] [7]. Part of this dataset was used to train the coverage prediction probability model, and other parts were used across the NANCY project.

Campus and experiment description

The experiments were conducted on two different days at the main campus of the University of Murcia (Espinardo Campus), which is covered by two distinct 5G private mobile-network operators (MNOs). Each operator provides overlapping service via two geographically separated cells:

- Operator Nokia
  - Cell 50 (Economicas site). Coord: 38.016969, -1.170034
  - Cell 51 (Pleiades site). Coord: 38.023726, -1.17311
- Operator AW2S
  - Cell 501 (Ática site): Coord: 38.022561, -1.174164.
  - Cell 502 (Luis Vives site): Coord: 38.016011, -1.172289.

During sampling Day 1, simultaneous samples from both networks were collected. In Day 2, only data from AW2S network were taken. The sampling routes start at the main door of the UMU's Computer Science Faculty and move around the campus, ensuring that each cell's coverage area was traversed multiple times (Figure 20).

Figure 20: Sampling Routes Map

5G Network Configuration

As mentioned, two different MNOs provide coverage to the UMU campus. Both networks operate at a bandwidth of 20 MHz and have two available cells, as depicted in Figure 20, AW2S in red and Nokia in blue. Nokia operates at 2586.050 MHz, while AW2S operates at 3435.00 MHz bands. Handover between cells is done in an intra-frequency way, which means that both cells of each operator are configured to work in the same frequency. Bearer used is default sst:1 and sd:1 with non-guaranteed bitrate mode. Nokia network is based on Frequency Division Duplex (FDD), while AW2S are based on Time Division Duplex (TDD).

Hardware and Device Configuration

A single monitoring unit was assembled as follows:

- **Host platform:** Industrial-grade embedded computer (x86) with Ubuntu 22.04 LTS, more concretely a LattePanda v3
- **Modems:**
  - **Nokia modem** (USB WWAN interface wwan0) Fibocom Fibocom FM160 connected to Operator Nokia
  - **AW2S modem** (USB WWAN interface wwan1) Fibocom Fibocom FM150 Modem_SN connected to Operator AW2S
- **GNSS receiver (optional):** USB GPS dongle, polled via gpsd for geo-referencing measurements

- **Power & mounting:** All hardware was rack-mounted inside a testing vehicle; antennas were roof-mounted for optimal signal (Figure 21)

Each modem was registered and configured using the libqmi-glib stack. The monitoring script was set to distinguish interfaces by name (wwan0 vs. wwan1) and tag each record with the corresponding operator.

Figure 21: Monitoring Unit - Testing Vehicle

Monitoring Script and Data Collection

A Python script [8] was developed that performs concurrent polling of both modems at fixed intervals, capturing:

1. **Network status & serving system** via QMI NAS calls
2. **Signal quality metrics** (RSRP, SINR, etc.) via QMI NAS signal-info calls
3. **Packet-data statistics** (counters and instantaneous channel rates) via Qualcomm MSM Interface Wireless Data Services (QMI WDS) calls
4. **Throughput measurements** by differencing cumulative byte-counters (via psutil) and converting to kbps
5. **GPS data** for spatial tagging

Key parameters:

- **Polling interval:** 1 s for all QMI calls
- **Throughput window:** 1 s, aligned to the QMI loop (both cumulative-delta and instant-rate computed over the same interval)

- **Logging format:** CSV with separate columns for each metric, plus timestamp.

Traffic characteristics

To test the connection performance, two different experiments were conducted. One is based on Transmission Control Protocol (TCP) traffic and the other on User Datagram Protocol (UDP) traffic. Both used the iPerf tool to send as much upload traffic as possible from the UE (iPerf client) to a service (iPerf server) instantiated in a common cloud. No other traffic was crossing the networks during the experiments.

Dataset organization

The datasets consist of 6 data files separated into two main folders (day 1 and day 2). In the file names, it is clearly stated the network and for what kind of traffic correspond the data in the file. The dataset structure is the following:

- Day 1:
  - tcp_aw2s_ 20240325.xlsx
  - tcp_nokia_ 20240325.xlsx
  - udp_aw2s_ 20240325.xlsx
  - udp_nokia_ 20240325.xlsx
- Day 2:
  - tcp_aw2s_ 20250126.xlsx
  - udp_aw2s_ 20250126.xlsx

Table 2 indicates the collected parameters and their description.

Table 2: Dataset Features

Category	Parameter	Value
Serving System	nas_value_serving_system_registration_state	Registration state of the modem (e.g., not registered, registering, registered)
Serving System	nas_value_serving_system_cs_attach_state	Circuit-switched (voice) attach state (attached/detached)
Serving System	nas_value_serving_system_ps_attach_state	Packet-switched (data) attach state (attached/detached)
Serving System	nas_value_serving_system_selected_network	Which Radio Access Technology (RAT) the device is currently camped on (e.g., UMTS, LTE, NR5G)
Serving System	nas_value_serving_system_radio_interfaces	List/bitmask of available radio interfaces on the serving system
Roaming & PLMN	nas_value_roaming_indicator	Indicates if the device is roaming (domestic vs. roaming)
Roaming & PLMN	nas_value_current_plmn_mcc	Current Mobile Country Code (numeric)

Roaming & PLMN	nas_value_current_plmn_mnc	Current Mobile Network Code (numeric)
Roaming & PLMN	nas_value_current_plmn_description	Human-readable operator name (PLMN description)
Legacy Signal Strength	nas_value_signal_strength_strength	Aggregate RSSI or signal-strength indication (unit depends on RAT)
Legacy Signal Strength	nas_value_signal_strength_radio_interface	The RAT/interface this strength refers to
Legacy Signal Strength	nas_value_io	Interference-over-noise metric (for CDMA/H-DR)
Legacy Signal Strength	nas_value_sinr	Signal-to-interference-plus-noise ratio (dB)
Legacy Signal Strength	nas_value_5g_signal_strength_rsrp	5G Reference Signal Received Power (RSRP) in dBm
Legacy Signal Strength	nas_value_5g_signal_strength_snr	5G Signal-to-Noise Ratio (dB)
Legacy Signal Strength	nas_value_5g_signal_strength_extended	Vendor-specific extended 5G strength metric
Rx-Chain Diversity	nas_value_rx_chain_{0–3}_info_rx_power	Per-chain (0–3) receive power
Rx-Chain Diversity	nas_value_rx_chain_{0–3}_info_ecio	Per-chain (0–3) Ec/Io
Rx-Chain Diversity	nas_value_rx_chain_{0–3}_info_rscp	Per-chain (0–3) RSCP
Rx-Chain Diversity	nas_value_rx_chain_{0–3}_info_rsrp	Per-chain (0–3) RSRP
Rx-Chain Diversity	nas_value_rx_chain_{0–3}_info_phase	Per-chain (0–3) antenna phase angle
Home-Network Info	nas_value_home_network_mcc	Home PLMN Mobile Country Code
Home-Network Info	nas_value_home_network_mnc	Home PLMN Mobile Network Code
Home-Network Info	nas_value_home_network_description	Home operator name
Home-Network Info	nas_value_network_name_source	How the network name was derived
NR5G Cell Details	nas_value_nr5g_arfcn	5G Absolute Radio Frequency Channel Number
NR5G Cell Details	nas_value_nr5g_cell_information_global_cell_id	Global Cell Identifier (NCGI)
NR5G Cell Details	nas_value_nr5g_cell_information_physical_cell_id	Physical Cell ID (PCI)
NR5G Cell Details	nas_value_nr5g_cell_information_plmn	PLMN of the serving NR5G cell
NR5G Cell Details	nas_value_nr5g_cell_information_rsrp	NR5G RSRP

NR5G Cell Details	nas_value_nr5g_cell_information_rsrq	NR5G RSRQ
NR5G Cell Details	nas_value_nr5g_cell_information_snr	NR5G Signal-to-Noise Ratio
NR5G Cell Details	nas_value_nr5g_cell_information_tracking_area_code	Tracking Area Code (TAC)
Radio Interfaces & Bands	nas_list_radio_interface / nas_extended_list_radio_interface	Supported/available radio interfaces
Radio Interfaces & Bands	nas_list_active_band_class / nas_extended_list_active_band_class	Active band classes
Radio Interfaces & Bands	nas_list_active_channel / nas_extended_list_active_channel	Active channel numbers
Radio Interfaces & Bands	nas_bandwidth_list_radio_interface	Interfaces for which bandwidth is reported
Radio Interfaces & Bands	nas_bandwidth_list_bandwidth	Reported bandwidth per interface
Packet-Data Statistics	wds_value_tx_packets_ok / wds_value_rx_packets_ok	Successfully sent/received packet counts
Packet-Data Statistics	wds_value_tx_packets_error / wds_value_rx_packets_error	Packet error counts
Packet-Data Statistics	wds_value_tx_overflows / wds_value_rx_overflows	Overflow event counts
Packet-Data Statistics	wds_value_tx_bytes_ok / wds_value_rx_bytes_ok	Total bytes sent/received
Packet-Data Statistics	wds_value_tx_packets_dropped / wds_value_rx_packets_dropped	Dropped packet counts
Packet-Data Statistics	wds_value_channel_rates_channel_tx_rate_bps	Current TX rate (bps)
Packet-Data Statistics	wds_value_channel_rates_channel_rx_rate_bps	Current RX rate (bps)
Packet-Data Statistics	wds_value_channel_rates_max_channel_tx_rate_bps	Max TX rate since last reset
Packet-Data Statistics	wds_value_channel_rates_max_channel_rx_rate_bps	Max RX rate since last reset
Packet-Data Statistics	wds_value_connection_status	Data bearer status
Throughput	throughput_upload_kb / throughput_download_kb	KB transferred since last sample
Throughput	throughput_upload_speed_kbps / throughput_download_speed_kbps	Instantaneous kbps up/down
GPS Fix Data	gpsd_sky_datetime / gpsd_sky_timestamp	Sky-view timestamp
GPS Fix Data	gpsd_sky_hdop / gpsd_sky_pdop	Sky-view dilution of precision (HDOP/PDOP)
GPS Fix Data	gpsd_tpv_datetime / gpsd_tpv_timestamp	TPV fix timestamp

GPS Fix Data	gpsd_tpv_lat / gpsd_tpv_lon / gpsd_tpv_alt / gpsd_tpv_althea	Latitude, longitude, altitude, altitude error
GPS Fix Data	gpsd_tpv_epx / gpsd_tpv_epy / gpsd_tpv_epv	Position error estimates (meters)
GPS Fix Data	gpsd_tpv_speed / gpsd_tpv_eps	Speed and its error estimate

Data preprocessing

For training the coverage probability prediction model, NANCY opted to use the data collected during the first day of the sampling campaign. To this end, we cleaned the data and removed incomplete samples before initiating the training and transfer learning operations. More specifically, we performed the following operations:

1. **Data cleansing.** During this process, we make sure that the dataset contains valid data values. To this end, we scan the whole sample space and we remove any data containing letters, symbols and corrupted information that is non-readable by the models.
2. **Outlier removal.** Within this process, we remove the top 5% and the bottom 5% of the data samples in terms of their values. This eliminates data points that deviate significantly from the rest of the data distribution, as they can negatively impact the performance of a model.

3.3. Training with Open Data and NANCY Data

For the implementation of the MLP and the rest of the components, we used the Python programming language and leveraged the functionalities of the Keras library. As mentioned in section 3.1, the model training process consists of two phases: **(i)** the model training operation, which utilises the "LTE Dataset" that is publicly available; and **(ii)** the fine-tuning phase, which utilises the NANCY dataset, collected through UMU's testbed. Below, we discuss the details of each phase:

1. The model training process utilized input from the publicly available "LTE Dataset". More specifically, we formulated a training set that consisted of 32,000 data samples, with each sample representing a measurement in a specific time slot. Each data sample had 2 features: (i) the distance of the UE from the closest cell tower; and the (ii) UE's SNR measured at the corresponding timeslot. We split the dataset into the 80%-10%-10% pattern, meaning that 80% of the samples were used as the training set (25,600 samples), 10% of the samples were used as the validation set (3,200 samples) and 10% of the samples were used as the test set (3,200 samples). The model was trained to predict the RSRQ value of the UE, given the input data.
2. The model fine-tuning process, which utilised the trained model (taken from phase 1) and conducted a transfer learning operation, as described in section 3.1. For this purpose, we leveraged the UMU dataset that consisted of 2,998 data samples, with each sample corresponding to two features. The features were the same as in phase 1 of the training process (i.e., distance of the UE from the cell and the UE's SNR). We again split the dataset using the 80%-10%-10% pattern, as follows: The training dataset consisted of 2,398 samples, the validation set of 300 samples and the test set of 300 samples.

Figure 22 illustrates the MAE loss within the transfer learning process for 500 epochs. We observe that the model manages to lower the MAE to 0.44, which amounts to an average 4% error rate. We consider this outcome a very good result, especially considering the fact that both the training and the transfer learning process require less than 3 minutes to complete.

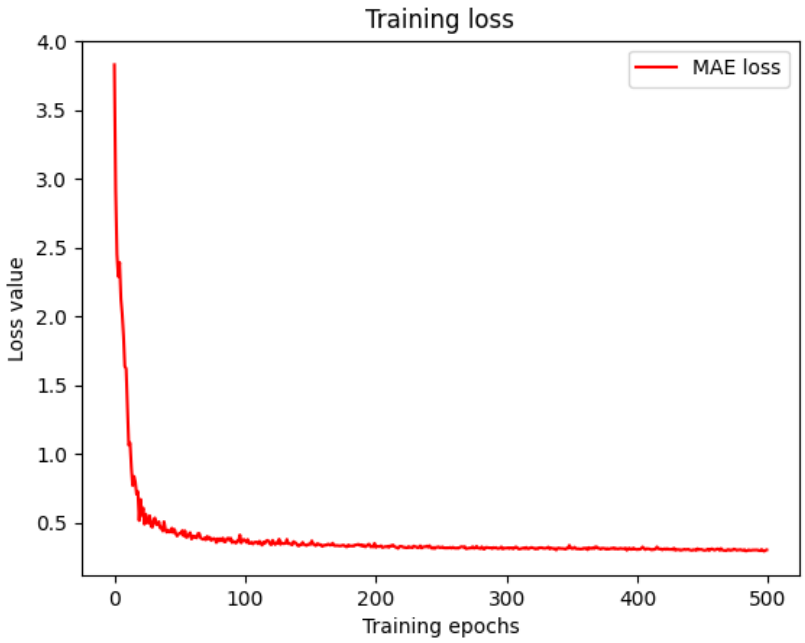

Figure 22. MLP training performance during the transfer learning process

4. Outage Probability Prediction

Network outage probability constitutes one of the most prominent performance measures for mobile cellular networks, as it reflects the overall network availability and its capability to provide seamless and ubiquitous coverage. B5G networks promise the provision of a wide range of services, each coming with unique requirements in terms of reliability, latency and throughput. The three main service categories in these networks, originally defined as Enhanced Mobile Broadband (emBB), Ultra Reliable and Low Latency Communications (URLLC) and Massive Machine Type Communications (mmTC) in 5G network services, also extend to the target performance requirements to B5G, namely xmBB, xURLLC and mmTC+ services. The formalization of these requirements in a more stringent manner extends from inheriting outage analysis models from 5G while assessing reliability variability, service ultra-low latency and supported extreme broadband data rate in massive deployments. Thus, next-generation networking services must adapt to dynamic channel conditions and steer the dynamic radio resource management process accordingly in decentralized architectures. Consistently satisfying those dynamic and heterogeneous service requirements is both crucial and challenging for both the network performance and end user experience. Defining a network outage event as an episode, where one or more of the performance metrics (e.g., throughput over the radio network) fails to satisfy a predetermined service-specific threshold, the occurrence of network outage events (interchangeably, service disruption events) must be minimized.

In particular, the second-order statistics of network outages, such as their frequency and duration, are directly related to several radio resource management tasks that aim to proactively or reactively cope with network outage events. These include the power allocation and the transmission rate choice (jointly making up what is called *link adaptation*), as well as other fundamental Radio Access Network (RAN) design choices such as the data block length that is transmitted over the air and the duration of transmission time slots [9].

Attempts to assess/predict the network outage probability in literature are related to both power-limited and interference limited regimes [10] and are tightly coupled with fading channel models [11]. Outage probability evaluation proves to be an essential task in the context of multipath diversity and correlated fading channels in cases where traditional channel models fail to capture [12] realistic cellular environments and the interference-limited assumption in ultra-dense deployments, as is the case in B5G networks. In turn, interference in mobile cellular networks is also dependent on the higher frequency bands [13] to be used in B5G networks with the aim of spectral efficiency. Traditional model-based and probabilistic [14] [15] approaches to outage probability computation call for numerical evaluations. Moreover, the closed-form expressions of outage probability are often too complex and need to be generalized to decentralized topologies and edge intelligent use cases.

A model-based approach to outage probability prediction remains tightly coupled with the channel estimation stage, thereby increasing complexity as B5G dynamic topologies and time-varying channel conditions emerge in the model creation step. This model turns out to be an essential prerequisite for optimizing resource allocation and improving user experience. On the contrary, an ML-powered model can circumvent the above series of steps, drastically reducing the complexity of prediction. The model can cater for diverse network features and leverage data to efficiently learn which of those are relevant and to what extent for predicting rate outages according to derived probability [16] [17].

4.1. Model Design and Methodology

4.1.1. Model choice and justification

In the NANCY context, the Outage Probability Prediction module is based on XGBoost (eXtreme Gradient Boosting), a distributed, open-source ML library that uses gradient-boosted decision trees. Decision trees [18] are primarily used for classification or regression tasks in ML. They exhibit a hierarchical structure: an internal node represents a feature, a branch represents a decision rule, and each leaf node represents the outcome of the dataset.

Boosting is an instance of so-called ensemble methods, which are used to create models with higher robustness to overfitting. Boosting [19] combines multiple individual weak trees, i.e. models that perform slightly better than random chance, to form a strong learner. Each such weak model is sequentially trained to correct the errors made by the previous models. After hundreds of iterations, weak learners are converted into strong learners.

XGBoost is known to exhibit certain advantages that make it one of the most popular ML algorithms, especially for structured/tabular data:

- Good predictive performance: XGBoost consistently wins or places highly in ML competitions (e.g., Kaggle), often outperforming other algorithms like random forests, Support Vector Machines (SVMs), and traditional gradient boosting.
- Built-in regularization capabilities: it includes L1-regularization (Lasso) and L2- regularization (Ridge) in the objective function, which help prevent overfitting, a common issue in traditional boosting methods.
- Built-in handling of missing values: the algorithm automatically learns how to handle missing data during its training process.
- Flexibility: it supports custom loss functions and evaluation metrics and can be adapted to be used for classification, regression, and ranking tasks with controllable time complexity
- Good scalability properties: the algorithm lends to efficient implementations leveraging data structures and parallel processing capabilities in the generation of trees and the parsing of features.

In addition to these generic advantages, when compared to logistic regression, SVMs, K-Nearest Neighbors, Random Forests, and Gradient Boosting, XGBoost has emerged as the model with the best performance overall, in terms of metrics considered in our module, in respect to similar classification tasks provided publicly available datasets, as in [20], [21], [22].

4.1.2. Model implementation

The model was implemented using the Python statistical and ML library scikit-learn. The input feature set included the following variables:

- dl_buffer [Amount of data in the downlink buffer]
- tx_pkts downlink [Number of packets transmitted in the downlink]
- dl_cqi [Downlink channel quality index]
- sum_requested_prbs [Total number of Physical Resource Blocks (PRBs) requested by UE]
- sum_granted_prbs [Total number of Physical Resource Blocks (PRBs) granted to UE]

Moreover, being directly related to throughput, these features, which are taken from Colosseum ORAN COMMAG dataset, can be categorized into two groups: the one, comprising dl_cqi, which constitutes a channel quality indicator, and tx_pkts_downlink, reflects radio link quality, which in turn can lead to rate outage events; whereas, the second category, including dl_buffer, sum_requested_prbs and sum_granted_prbs, captures the level of traffic demand in the network. Both delineate the interplay of environment and networking parameters influencing throughput that is achieved and can capture the complexity of the rate outage events' occurrence conditions in the network, from both of the aforementioned perspectives. In this context, the XGBoost model was applied for predicting rate outage events, utilizing these features, which were defined above.

The variable used for determining rate outages was the downlink throughput (tx_brate downlink) from the BS to the UE. As part of the model pre-processing, the continuous throughout values are converted to binary values, 0 or 1, denoting network availability or outage, respectively. In this respect, two types of thresholds R_{thr} were considered:

- Network service-related thresholds: the outage threshold for URLLC is considered equal to 0.01 Megabits per second (Mbps) and for MTC (Machine Type Communications) equal to 0.03 Mbps. These service-related thresholds were derived from the Colosseum ORAN COMMAG [23].
- The statistical and/or experimental-based thresholds: these are statistics computed directly out of the throughput values in the datasets themselves, namely the average threshold equal to 0.014 Mbps and 25th percentile threshold equal to 0.006 Mbps.

As part of the post-processing phase, the output values of the model (probability values in [0,1]) were converted to binary variables, using a threshold *p*. Experimenting with different *p* values in [0.1, 0.5], we found the best scores with *p* = 0.2. Namely, we term an event an outage if the output probability generated by the model (i.e., the probability that the rate falls below R_{thr}) exceeds the threshold probability value of 0.2. This choice makes the model more pessimistic (or conservative) as to when outage events are declared, i.e., the model will maybe declare more outage events, reducing false negative events (a non-detected outage) at the expense of more false positives (declare outage events that do not actually qualify as outages).

4.2. Data Collection

The data for the training of the AI Network Quality Module (AINQM) came from the publicly available Colosseum dataset (Colosseum O-RAN COMMAG Dataset) [23], a dataset that has been widely acclaimed for its significant size as well as its variety of features. Its name stems from the ancient stadium in Rome, where a massive 5G experiment took place, involving 40 pieces of UEs and 4 BSs. The whole Colosseum dataset is organized hierarchically according to how UEs are divided into traffic-dependent slices, their mobility speeds, their distribution around the BS, as well as the algorithm used by the RAN scheduler to serve them. The dataset used in these experiments contains 1824 tuples (X, y), each carrying values for the feature set X used as input to the model and the target variable y (i.e., the probability that the downlink throughput will fall below R_{thr}).

The types of model features and their range of values are as follows:

- the dl_buffer is an integer number ranging from 0 to 459 bytes
- the tx_pkts is an integer value ranging from 0 to 103 packets
- the dl_cqi is a float number varying in [0.1..15]

- the sum_requested_prbs is an integer value varying from 0 to 605
- the sum_granted_prbs is an integer value that varies from 0 to 568

4.3. Training with Open Data

For the model training process, the dataset was split into a training data part (90% of the dataset) and a testing data part (the other 10%).

As formerly indicated, the Outage Probability Prediction model is firmly based on the definition of network downlink rate outage. As stated above, four threshold values were considered, two values per threshold type mentioned in section 4.1.2. Among those, our experimentation-based quantile threshold generated the best results when one jointly considers the standard metrics listed below:

- $\Pr e\,cision = \frac{TruePositives}{TruePositives + FalsePositives}$; a measure of XGBoost model predicting positive instances. This metric is defined as the ratio of true positive predictions to the number of both true and false positive predictions.
- $Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$; it assesses the performance of the classifier to identify positive instances from the actual positive instances contained in the dataset.
- $F1 - score = \frac{\Pr ecision \cdot Recall}{\Pr ecision + Recall}$; this metric assesses how well the Xgboost model in terms of balancing precision and recall and is indicative of how efficiently false positives and false negatives are reduced.
- $Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives}$; it is defined as the ratio of correct predictions to the total number of predictions and is a measure of outage probability prediction performance

5. Latency Prediction

In the context of B-RAN environments, the latency of the network is directly correlated with the latency experienced during blockchain operations. This is influenced by both network conditions and the internal complexity of the consensus protocol. Different blockchain consensus protocols have varying complexities that directly impact their operational latency.

Understanding and predicting the latency behaviour of these protocols is essential for optimizing network performance and resource allocation in B-RAN systems. In this work, we apply an AutoRegressive Integrated Moving Average (ARIMA) model to forecast blockchain operation latency, providing a lightweight yet effective approach to predictive performance management across varying protocol complexities.

5.1. Model Design and Methodology

5.1.1. Blockchain consensus mechanisms overview

In this section, we briefly explore several consensus mechanisms used in blockchain systems, focusing on their communication complexity and security characteristics. These protocols form the backbone of distributed agreement in fault-prone environments, and understanding their trade-offs is essential for evaluating the impact they will have on the network latency.

Table 3 summarizes key properties of four consensus protocols, Reliable, Replicated, and Fault Tolerant (RAFT) consensus algorithm [24], Quorum-based Byzantine Fault Tolerance (QBFT) [25], Istanbul Byzantine Fault Tolerance (IBFT) [26] and Scalable Byzantine Fault Tolerance (SBFT) [27], used in blockchain systems, comparing their message complexity and resilience to attacks (in terms of the number and type of faulty nodes tolerated).

Table 3: Blockchain Consensus Protocols Summary

Protocol	Type	Message Complexity	Fault Tolerance
RAFT	Crash Fault Tolerant (CFT)	$O(n)$	Tolerates up to $\lfloor (n-1)/2 \rfloor$ crash faults
QBFT	Byzantine Fault Tolerant (BFT)	$O(n^2)$	Tolerates up to $\lfloor (n-1)/3 \rfloor$ Byzantine faults
IBFT	BFT	$O(n^2)$	Tolerates up to $\lfloor (n-1)/3 \rfloor$ Byzantine faults
SBFT	Scalable BFT	$O(n)$	Tolerates up to $\lfloor (n-1)/3 \rfloor$ Byzantine faults

The variable n refers to the number of nodes (or validators) participating in the protocol. These nodes are responsible for proposing, validating, and agreeing upon transactions or blocks in the network. The performance and fault tolerance of a protocol are analyzed in terms of n, as it directly influences the communication overhead and the number of faults the system can withstand.

Fault tolerance in consensus protocols refers to the system's ability to continue operating correctly despite failures. These failures may be:

- Crash faults, where a node simply stops responding.
- Byzantine faults, where a node behaves arbitrarily or maliciously, potentially sending false or contradictory messages.

RAFT is designed to handle only crash faults. It assumes that nodes either operate correctly or fail by becoming unresponsive. If the majority of nodes remain honest and available, the system continues to function.

QBFT, IBFT, and SBFT are all designed to handle Byzantine faults, which include both crashes and malicious actions. They rely on strict quorum sizes, typically requiring agreement from at least two-thirds of nodes, to ensure that honest nodes overlap and prevent conflicting decisions. Combined with cryptographic techniques, these quorum requirements enable consensus even in the presence of faulty or adversarial nodes. This higher level of assurance comes with increased communication complexity.

BFT

Byzantine Fault Tolerance is a foundational concept in distributed systems, especially in adversarial environments like blockchains. A BFT system can reach agreement among distributed nodes even when some of those nodes are malicious or compromised. The name originates from the "Byzantine Generals Problem" [28], where actors must agree on a common strategy despite some of them being traitors.

BFT protocols typically work by requiring multiple rounds of message exchange to ensure that all non-faulty nodes agree on the same values. These rounds include stages like proposal, prepare, commit, and sometimes view change. To withstand f Byzantine faults, the protocol needs at least$n = 3f + 1$ nodes. This ensures that the number of honest nodes is always greater than the potential coalition of malicious actors.

Modern adaptations of BFT, like QBFT and SBFT, introduce optimizations such as partial signature aggregation, collectors, and threshold cryptography to reduce the message complexity while maintaining the same level of security. These improvements are essential for deploying BFT systems in large-scale, performance-sensitive blockchain environments.

QBFT

QBFT is a variant of classical BFT consensus protocols that emphasizes strong safety and liveness guarantees in adversarial environments. QBFT proceeds through multiple communication rounds, typically pre-prepare, prepare, and commit, where each node communicates with every other node to achieve consensus.

This communication pattern results in a message complexity of $O(n^2)$, as every node must exchange messages with all others during each phase of the protocol. While it includes some optimizations, such as batched signatures and quorum-based agreement, these measures help with practical performance but do not reduce the theoretical communication complexity.

IBFT

IBFT is a practical BFT consensus algorithm. It follows a three-phase commit protocol: pre-prepare, prepare, and commit. In each phase, every node must communicate with all other nodes to exchange and verify messages.

Due to this all-to-all communication pattern in every round, the message complexity of IBFT is $O(n^2)$. This redundancy provides strong fault tolerance and safety guarantees even in the presence of Byzantine nodes, but it also limits scalability and increases latency as the number of nodes grows.

SBFT

SBFT is designed to address the scalability limitations of traditional BFT protocols. SBFT introduces mechanisms like collectors and threshold cryptography, which allow subsets of nodes to aggregate

messages and signatures efficiently. These features reduce the number of messages each node must send and receive.

As a result, SBFT achieves amortized $O(n)$ message complexity. Although individual operations may sometimes require more communication, the average per-operation cost remains linear. This makes SBFT especially suitable for high-throughput applications where maintaining low latency and communication overhead is crucial.

RAFT

RAFT is a consensus algorithm built for simplicity and fault tolerance in distributed systems where nodes may crash but are not malicious. The protocol elects a leader node which handles all client interactions and coordinates the replication of log entries to follower nodes. When a majority of followers acknowledge an entry, it is considered committed.

The protocol achieves its consensus with a message complexity of $O(n)$ because each operation involves the leader sending messages to all other nodes (followers) and receiving acknowledgements. The simplicity and centralization around a leader reduce overhead but also mean the system is only resilient to crash faults, not Byzantine behaviours.

5.1.2. Model selection

Before diving into the mechanics of the ARIMA model, it's important to highlight one of its key advantages in the context of B-RAN systems: it requires no explicit training phase. Instead, the ARIMA model adapts dynamically to incoming data, making it highly suitable for real-time forecasting tasks in changing network environments [29]. This adaptability allows the model to be applied flexibly across different B-RAN configurations, regardless of the network's architecture.

Furthermore, we can leverage theoretical insights into the complexity of consensus algorithms used by protocols such as RAFT, QBFT, IBFT, and SBFT. These algorithms often impose computational or communication costs that grow linearly, quadratically, or even cubically with the number of orderers. While ARIMA itself is agnostic to the exact functional form of the latency growth, its differencing and autoregressive structure can adapt to these changes, effectively capturing and predicting the underlying trend [30]. Furthermore, we can create dummy data that follow those specific trends to predict our model's performance.

It is also worth noting that in the relatively short and constrained timeframes, ranging from ms to a few seconds, typically relevant to operational B-RAN scenarios, latency timeseries do not generally exhibit seasonality. As such, a non-seasonal ARIMA model is appropriate, simplifying the modelling process without sacrificing accuracy.

Understanding the ARIMA(p, 1, 5) Model

The ARIMA model we use is defined by three parameters: the order of autoregression (p), the degree of differencing (d = 1), and the order of the moving average (q = 5).

Differencing is a preprocessing step designed to remove trends and make the data more stationary. A time series is said to be stationary if its statistical properties, like mean and variance, are constant over time. Stationarity is a desirable property because most time series models, including ARIMA, perform best when the input series does not have trends or seasonality.

Given an original time series y_t, we compute the first-order differenced series z_t as:

$$z_t = y_t - y_{t-1}$$

This operation effectively removes linear trends from the original series. We then model the differenced series z_t using a combination of past values of z (the autoregressive component) and past forecast errors (the moving average component).

Mathematically, the model is expressed as:

$$z_t = \sum_{i=1}^{p} \Phi_i \, z_{t-1} + \sum_{j=i}^{5} \theta_j \, \varepsilon_{t-j} + \varepsilon_t$$

Here, Φ_i are the autoregressive coefficients, θ_i are the moving average coefficients, and ε_t is the white noise error at time t. The goal of the model is to estimate the next differenced value z_{t+1} using a weighted sum of previous values and residuals.

Once we have predicted z_{t+1}, we return to the original scale of the time series by inverting the differencing operation:

$$\overline{y}_{t+1} = y_t + \overline{z}_{t+1}$$

This gives us a forecast of the next actual value in the original series.

To estimate the autoregressive coefficients $\Phi_1, \ldots, \Phi_p$, we must solve a system of equations derived from the autocorrelation function of the differenced series z_t. These are known as the Yule–Walker equations and can be expressed in matrix form as:

$$R\Phi = r$$

Where R is a symmetric Toeplitz matrix composed of autocorrelation values $r_k = E[z_t z_{t-k}]$, Φ is the vector of autoregressive coefficients, and r is the autocorrelation vector.

Instead of directly solving this system using computationally expensive matrix inversion, we apply the Levinson-Durbin recursion. This is a recursive algorithm that builds the solution incrementally from order 1 up to order p. At each step, it computes a reflection coefficient (also called the partial autocorrelation) and updates the prediction coefficients based on this value.

The key intuition behind Levinson-Durbin is that it efficiently finds the best linear predictor for a stationary process using its autocorrelation structure. By avoiding explicit matrix inversion and instead leveraging the structure of the Toeplitz matrix, the algorithm achieves a much lower computational cost,$O(p^2)$, which is important for real-time or embedded applications.

In the context of ARIMA, Levinson-Durbin is used after differencing has been applied. At this stage, the series z_t is assumed to be stationary, and we aim to model it using an autoregressive process. Levinson-Durbin takes the autocorrelation structure of z_t and computes the coefficients Φ_i that describe how each value of z_t depends on its previous values. These coefficients are essential in generating the AR component of the ARIMA model. Once these are known, they can be used to forecast the next value in the differenced series, which is then integrated to produce a forecast in the original scale.

This integration of Levinson-Durbin into ARIMA provides both theoretical robustness and computational efficiency. It ensures that the ARIMA model has a strong foundation in statistical theory while remaining practical for application to real-world datasets, especially when computational resources are limited or when quick forecasting is required.

This approach combines simplicity and flexibility. Differencing removes persistent trends, making the modelling problem easier. The autoregressive part captures longer-term dependencies in the data.

The moving average component helps smooth out short-term noise or unexpected fluctuations, improving robustness and prediction stability. By estimating the autoregressive part using Levinson-Durbin, we gain numerical efficiency and stability, especially when the number of observations is small or when the autocorrelation matrix is ill-conditioned. This method is best suited for univariate time series data that exhibit both trend and irregular noise but do not have complex seasonal patterns.

5.2. Data Collection

5.2.1. Environment setup

The data creation process includes a combination of internally generated experimental data and simulated workloads designed to emulate real-world blockchain environments. The primary focus was to analyse the latency and throughput characteristics of various blockchain systems and consensus protocols, specifically the ones mentioned in Table 4:

Table 4: Systems Under Test (SUTs)

Platform	Version	Consensus Protocol
Hyperledger fabric [31]	V2.5	Raft (CFT) [32]
	V3.0	SBFT [33]
Hyperledger Besu [34]	V25.0	QBFT [35]
		IBFT2.0 [36]

These decentralised networks were deployed in a controlled environment that allowed to monitor, record, and extract key performance metrics under different configurations and workloads. The data collection focused on network latency, block propagation times, transaction finality and overall system responsiveness under various stress levels. The experiment was conducted on blockchain solutions that are appropriate for industrial use cases and are in close comparison with the one used by NEC's proposed architecture within NANCY. The tool that was used for conducting the tests was Hyperledger Caliper [37], an open-source benchmarking tool designed to evaluate the performance of blockchain networks. It allows users to run predefined or custom test workloads on various blockchain platforms and generates detailed reports on key performance metrics such as transaction latency, throughput (TPS), and success/failure rates.

The characteristics of the workstation that was used for the benchmarking process is described in Table 5.

Table 5: Workstation specifications

Component	Specification
Operating System	Ubuntu 24.04.2 LTS
Motherboard	Gigabyte Z790 AORUS ELITE AX
CPU	Intel® Core™ i7-14700KF (20-core: 8 P-cores + 12 E-cores, up to 5.5 GHz)
Memory	64 GiB DDR5 RAM (2×32 GiB @ 5600 MHz)

5.2.2. Benchmark configuration

Figure 23 is an example of a benchmark configuration file that was used for the tests.

```
test:
  workers:
    number: 10
  rounds:
    - label: Fixed-rate 100.
      txDuration: 60
      rateControl:
          type: fixed-rate
          opts:
            tps: 100
      workload:
        module: benchmarks/scenario/simple/fabcar/createCar.js
```

Figure 23: Benchmark config file

Each key parameter corresponds to a specific value parameter, with the key - value pairs explained Table 6.

Table 6: Benchmark configuration explanation

Parameter	Description
test.workers	Configuration for how transaction load is generated
test.workers.number	Specifies the number of worker processes to use for executing the workload
test.rounds	Array of objects, each describing the settings of a round
test.rounds[i].label	A short name of the rounds, usually corresponding to the types of submitted TXs.
test.rounds[i].txDuration	The length of the round in seconds during which Caliper will submit TXs.
test.rounds[i].rateControl	The object describing the rate controller to use for the round
test.rounds[].rateControl.type	The rate control strategy. fixed-rate means transactions are sent at a steady, constant rate.
test.rounds[].rateControl.opts.tps	TPS to send. Here, Caliper will attempt to submit 100 transactions per second throughout the 60-second test
test.rounds[i].workload	The object describing the workload module used for the round
test.rounds[i].workload.module	The path to the benchmark workload module implementation that will construct the TXs to submit

A configuration with the above structure will define a benchmark run that consists of multiple rounds. Each round is associated with a rate controller that is responsible for the scheduling of transactions (TXs) and a workload module that will generate the actual content of the scheduled TXs.

The benchmarking included various transaction rates, ranging from 100 to 1300 TPS in fixed steps (Table 7). Each test round lasted 60 seconds, with 10 local worker threads generating load. The same

application workload was used across all tests. The standard FabCar smart contract was used, simulating realistic asset creation scenarios. Also, different network configurations have been used with different numbers of orderers and validators ranging from 3 to 20 for the end results to be comparable with the NEC's proposed consensus algorithm and their testing results. The controller of choice was fixed rate to simulate a stable transaction load and to mimic a high but stable demand of transaction executions. The fixed-rate controller sends transactions at a constant, pre-defined rate (i.e., a fixed number of TPS).

Table 7: Benchmark round parameters

Parameter	Description
Benchmarking tool	Hyperledger Calieper v0.6
Transaction Rates (TPS)	100 to 1300TPS (in fixed steps of 100)
Test duration	60 seconds per round
Load Generation	10 local workers
Workload	createCar transaction for the FabCar smart contract
Network configuration	Varying number of orderers/validators depending on protocol (ranging 3 to 20)

5.2.3. Workload generation setup

A sample of the workload module used for the testing is provided in Figure 24:

```
'use strict';

const {WorkloadModuleBase} = require('@hyperledger/caliper-core');

const colors = ['blue', 'red', 'green', 'yellow', 'black', 'purple', 'white', 'violet', 'indigo', 'brown'];
const makes = ['Toyota', 'Ford', 'Hyundai', 'Volkswagen', 'Tesla', 'Peugeot', 'Chery', 'Fiat', 'Tata', 'Holden'];
const models = ['Prius', 'Mustang', 'Tucson', 'Passat', 'S', '205', 'S22L', 'Punto', 'Nano', 'Barina'];
const owners = ['Tomoko', 'Brad', 'Jin Soo', 'Max', 'Adrianna', 'Michel', 'Aarav', 'Pari', 'Valeria', 'Shotaro'];

/**
 * Workload module for the benchmark round.
 */
class CreateCarWorkload extends WorkloadModuleBase {
    /**
     * Initializes the workload module instance.
     */
    constructor() {
        super();
        this.txIndex = 0;
    }

    /**
     * Assemble TXs for the round.
     * @return {Promise<TxStatus[]>}
     */
    async submitTransaction() {
        this.txIndex++;
        let carNumber = 'Client' + this.roundArguments.initialId + this.workerIndex + '_CAR' + this.txIndex.toString();
        let carColor = colors[Math.floor(Math.random() * colors.length)];
        let carMake = makes[Math.floor(Math.random() * makes.length)];
        let carModel = models[Math.floor(Math.random() * models.length)];
        let carOwner = owners[Math.floor(Math.random() * owners.length)];

        let args = {
            contract: 'Fabcar',
            verb: 'createCar',
            args: [carNumber, carMake, carModel, carColor, carOwner],
            readOnly: false
        };

        await this.sutAdapter.sendRequests(args);
    }
}

/**
 * Create a new instance of the workload module.
 * @return {WorkloadModuleInterface}
 */
function createWorkloadModule() {
    return new CreateCarWorkload();
}

module.exports.createWorkloadModule = createWorkloadModule;
```

Figure 24: Workload module

Workload modules are implemented as Node.JS [38] modules that expose a certain Application Programming Interface (API). There are no further restrictions on the implementation, allowing developers to implement arbitrary logic (using further arbitrary components).

These modules are loaded through factory functions, just like other pluggable modules in Hyperledger Caliper. Accordingly, a workload module implementation must export a single factory function, named "*createWorkloadModule*". The factory function must return an instance that implements the [WorkloadModuleInterface]. The "*submitTransaction*" function is the backbone of the workload generation. The worker process calls this function every time the rate controller enables the next TX and is its responsibility to submit the TX through the connector API.

5.2.4. Key Performance Metrics

Hyperledger Caliper provides a detailed set of performance metrics after each benchmarking test, helping you evaluate how well a blockchain system performs under load. Table 8 summarizes the key metrics Caliper returns and their corresponding descriptions.

Table 8: Network performance metrics

Metric	Description
Success	Number of transactions that were successfully submitted and confirmed on the blockchain
Failed	Number of transactions that were submitted but **failed** due to errors like timeouts, endorsement issues, or network problems
Send Rate (TPS)	The average number of transactions **sent per second** during the test.
Throughput (TPS)	The number of **successful transactions per second** (i.e., how many were confirmed on-chain).
Latency (min/avg/max)	Time taken for a transaction to go from submission to confirmation. Caliper reports the **minimum**, **average**, and **maximum** latency values.

5.3. Training with mock Data

Although ARIMA requires no training in the traditional sense, we validated and fine-tuned its performance using mock data designed to replicate expected latency trends under different blockchain consensus complexities. Specifically, we generated synthetic datasets, consisting of 500 data points with linear and quadratic growth patterns, each affected by random noise relative to the signal strength (2%-5%). These datasets emulate how the number of transactions handled by the blockchain influences the latency over time.

In shorter time windows, the number of participating nodes is relatively stable, resulting in latency that aligns with a consistent linear-like computational pattern. However, over longer durations, network reconfigurations or load variations can cause this complexity to shift, reflecting the distinct scalability behavior of each consensus algorithm, either linear or quadratic, depending on the blockchain protocol (e.g. RAFT vs. IBFT).

To evaluate our ARIMA implementation, we applied it to these mock datasets and observed how its forecasting accuracy responded to different numbers of input points. As demonstrated in the MAPE (Mean Absolute Percentage Error) plots (Figure 25 & Figure 26), the model maintains reliable accuracy across increasing input sizes (input window) from 3-70, with error decreasing as more historical points are made available. We selected 60 input points (window size) as an effective balance between model performance and real-time feasibility. At this range, we observed from Figure 25 that increasing the window size beyond 60 points does not significantly reduce the error. A similar plateauing effect is visible in Figure 26 as well. This suggests that 60 input points represent a pragmatic compromise between the two trends, offering sufficient context for accurate forecasting without overfitting the data. As a result, the model is capable of generalizing effectively across both linear and nonlinear latency patterns while maintaining robustness against noise.

The accuracy of the ARIMA model at 60 input points is further confirmed by the predicted vs actual latency plots (Figure 27 & Figure 28) for both linear and quadratic mock datasets. In the linear case, the model closely tracks the increasing latency with a near-linear slope, achieving a MAPE of 0.029. The prediction line almost perfectly overlays the actual values, highlighting ARIMA's capacity to adapt to deterministic trends even in the presence of noise.

In the quadratic case, which emulates longer prediction timeframes and more complex consensus mechanisms like IBFT and SBFT, the model also performs well. The predicted curve adheres closely to the ground truth across a wide range of transaction counts, resulting in MAPE of 0.013. This shows

that the model is capable of capturing curvature in latency growth. These results collectively support the robustness and adaptability of ARIMA when used with a 60-point input window.

The dataset collected using the Hyperledger Caliper framework, as described in the previous section, will be used to test our model in 6.1.3, since it represents measurements taken from a simulation and are closer to a real-world scenario.

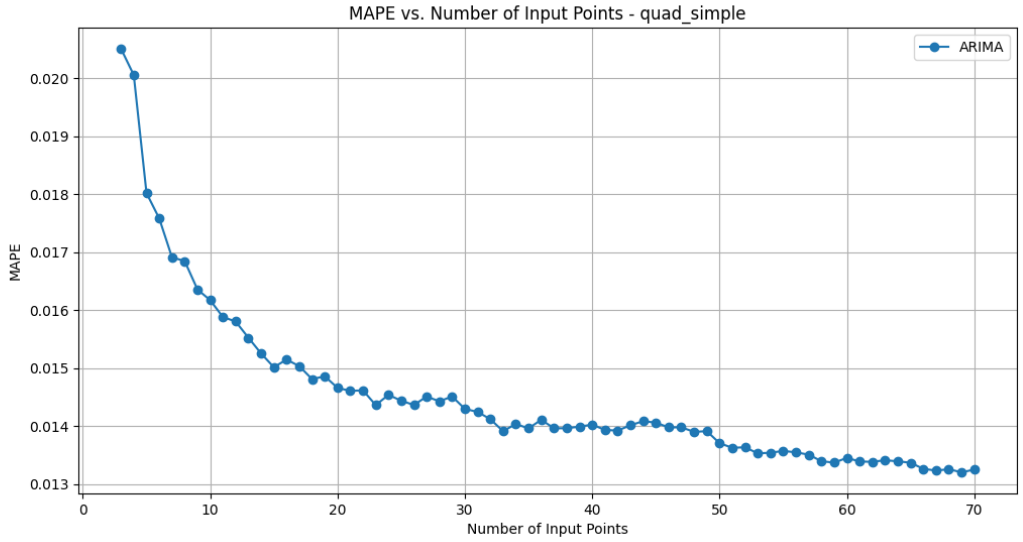

Figure 25: MAPE vs Number of input points – Quadratic data

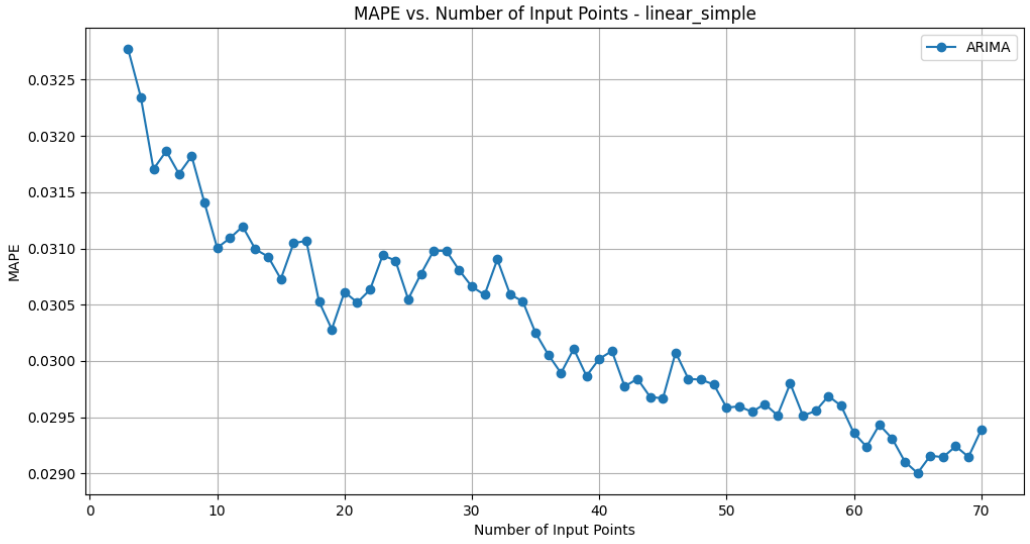

Figure 26: MAPE vs Number of input points – Linear data

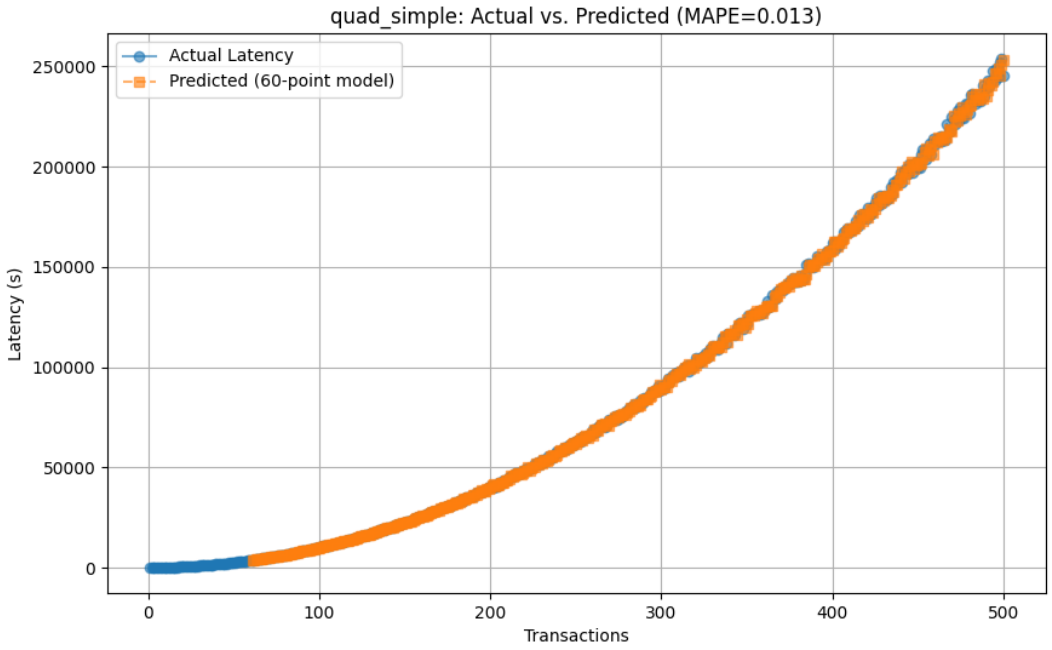

Figure 27: Real vs Predicted latency – Quadratic data

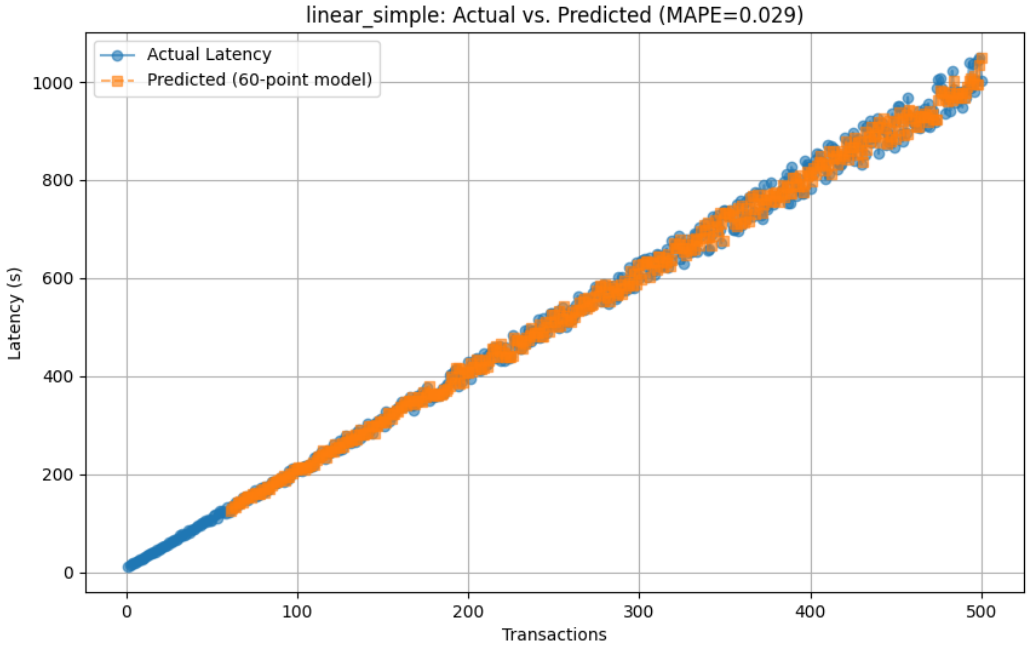

Figure 28: Real vs Predicted latency – Linear data

6. Performance Evaluation

6.1. Individual Model Evaluation

6.1.1. Coverage Probability Prediction Model

In order to properly evaluate the model for coverage probability prediction (see section 3), we utilise the real-world dataset collected by UMU's testbed. The dataset contains more than 7,300 data samples, collected over a 20-minute period that contain over 60 features. Our goal is to utilise the trained to model to predict the RSRQ value at any given time, which can then be used to assess the coverage probability of the UE. To accomplish this, we formulate a "test dataset" that consists of the data collected during the second day of the sampling Campaign. We prepare the dataset for inference by performing a data cleansing operation thus, removing invalid values but we refrain from removing outliers.

In order to evaluate our model's performance, we employ the MAE and MSE metrics. The MAE formula is given in section "3.1 Model Design and Methodology", while the MSE formula is calculated using the following equation:

$$MSE = \frac{\sum_1^n (y_i - x_i)^2}{\mathrm{n}}$$

Where (similar to MAE formula) y_i is the actual (real) value for data point *i*, x_i is the model output (prediction) and *n* is the total number of samples used for testing the model. Table 9 depicts the evaluation results. The model achieves 0.218 MSE and 0.137 MAE scores, meaning that its predictions diverge only by 0.137 (on average) from the real RSRQ values of the UE. This high accuracy is also accompanied by a very fast execution time (0.00054 seconds per prediction) which was expected due to the lightweight nature of the design.

Table 9: Coverage probability prediction model evaluation results

Evaluation metric	Score
MSE	0.218
MAE	0.137
Execution time	$5.4 * 10^{-4}$ seconds per prediction

Figure 29 illustrates the test predictions of the coverage probability prediction model along with the ground truth values for different distances (depicted in meters) between the UE and the Cell tower. As the UE moves away from the cell tower, its RSRQ values drop and thus its coverage probability also lowers. The model managed to capture this pattern and produced outputs that closely resemble the actual RSRQ levels of the UE.

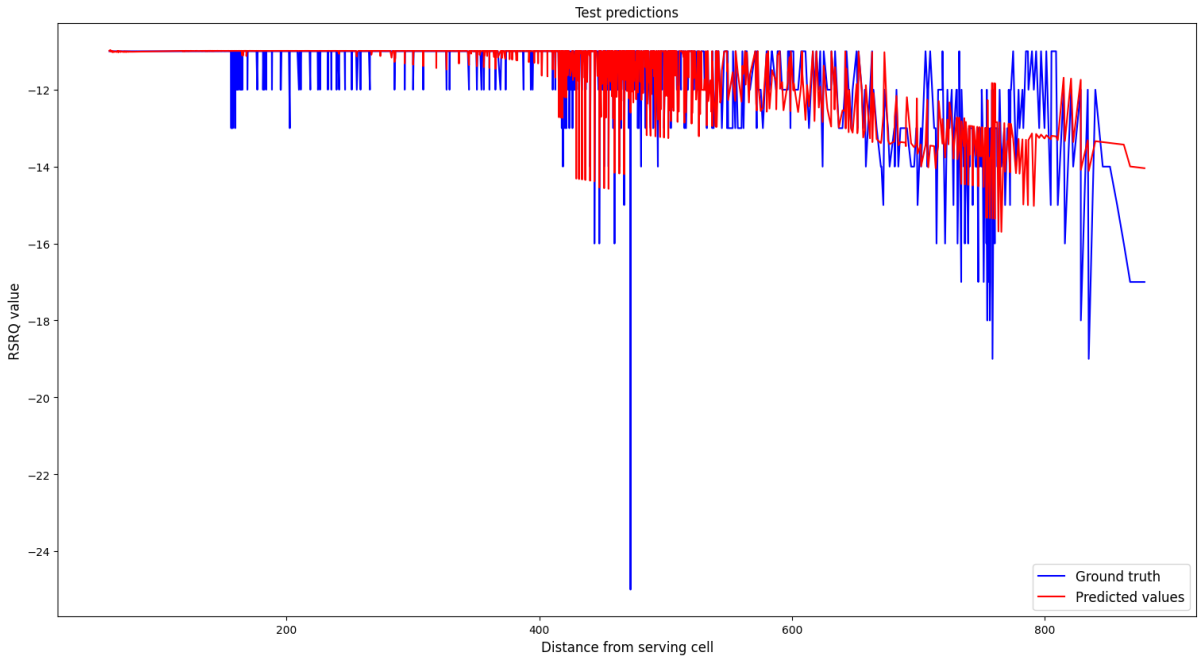

Figure 29. Test results of the coverage probability prediction model, over different UE-cell tower distances

6.1.2. Outage Probability Prediction Model

As formerly indicated, the Outage Probability Prediction model is strongly dependent on the definition of the network downlink rate outage. As described in section 4.1.2, four thresholds were considered to this end, and the model quality was assessed for each of these thresholds, jointly considering the standard metrics:

- Precision: A measure of XGBoost model predicting positive instances. This metric is defined as the ratio of true positive predictions to the number of both true and false positive predictions.
- Recall: It assesses the performance of the classifier to identify positive instances from the actual positive instances contained in the dataset.
- F1-score: It is defined as the ratio of correct predictions to the total number of predictions and is a measure of outage probability prediction performance; this metric assesses how well the XGBoost model performs in terms of balancing precision and recall and is indicative of how efficiently false positives and false negatives are reduced.
- Accuracy: It is defined as the ratio of correct predictions to the total number of predictions and is a measure of outage probability prediction performance.

Table 10 sumamrizes the various threshold values utilized for each one of the testing sessions for the AINQM module, along with the metrics that were derived from these sessions. It is quite apparent that the quantile experimental threshold outperforms, notwithstanding the solid performance of every other model, the rest of the models.

Table 10: XG Boost Performance Metrics for Four Thresholds Applied

Threshold	Accuracy	ROC-AUC	Precision	Recall	F1
0.014 (mean)	0.961	0.984	0.959	0.963	0.960
0.006 (quantile)	0.972	0.987	0.966	0.970	0.968
0.01 (URLLC)	0.95	0.972	0.945	0.942	0.943
0.03 (MTC)	0.987	0.988	0.872	0.872	0.872

The confusion matrices for each of the aforementioned four thresholds applied to determine outage events, along with the ROC curves, are provided in Figure 30, Figure 31, Figure 32 and Figure 33, clearly depicting XGBoost model performance for each of the four rate thresholds considered.

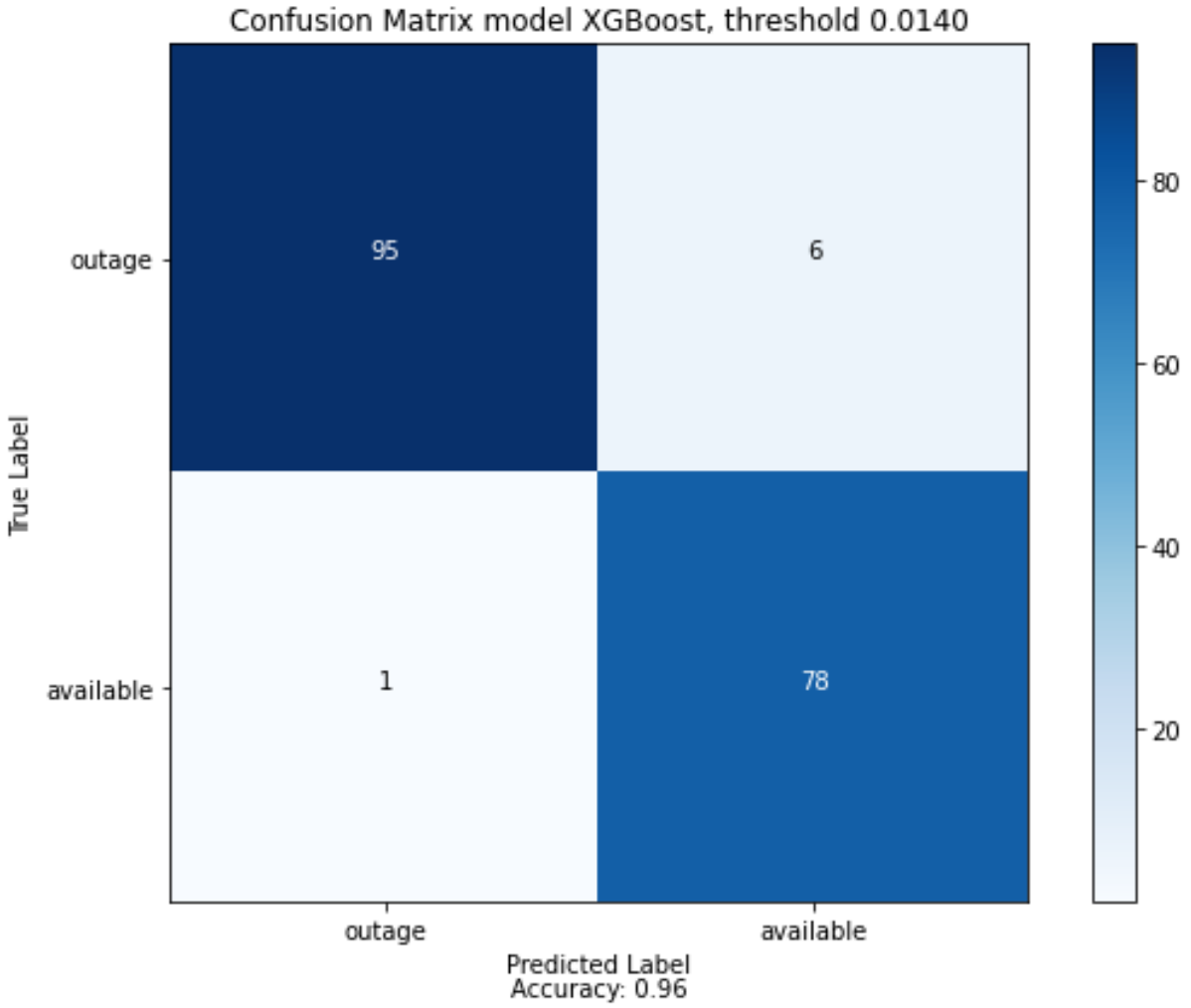

Figure 30: Confusion matrix for Average Threshold Case

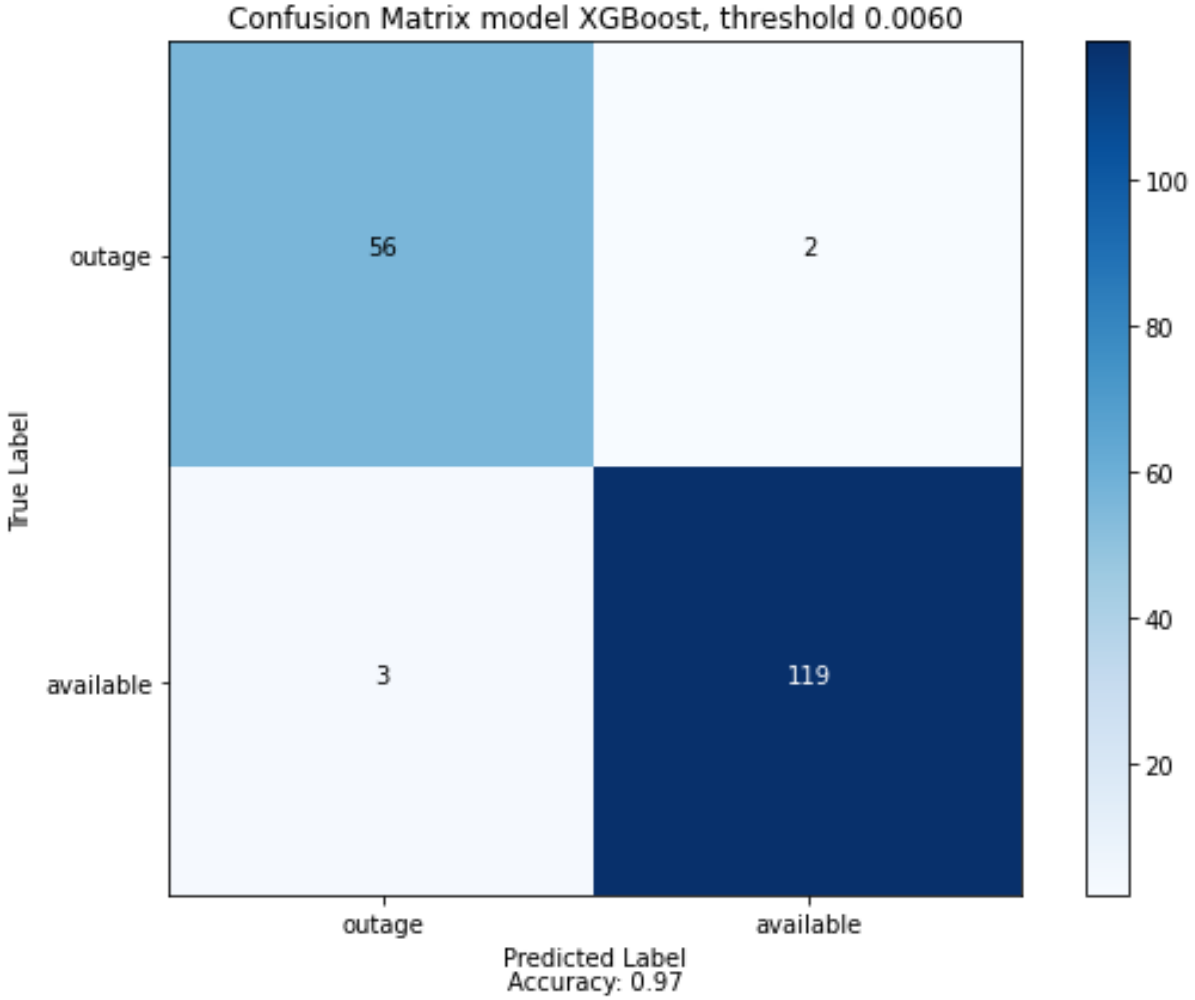

Figure 31: Confusion Matrix for Quantile Threshold Case

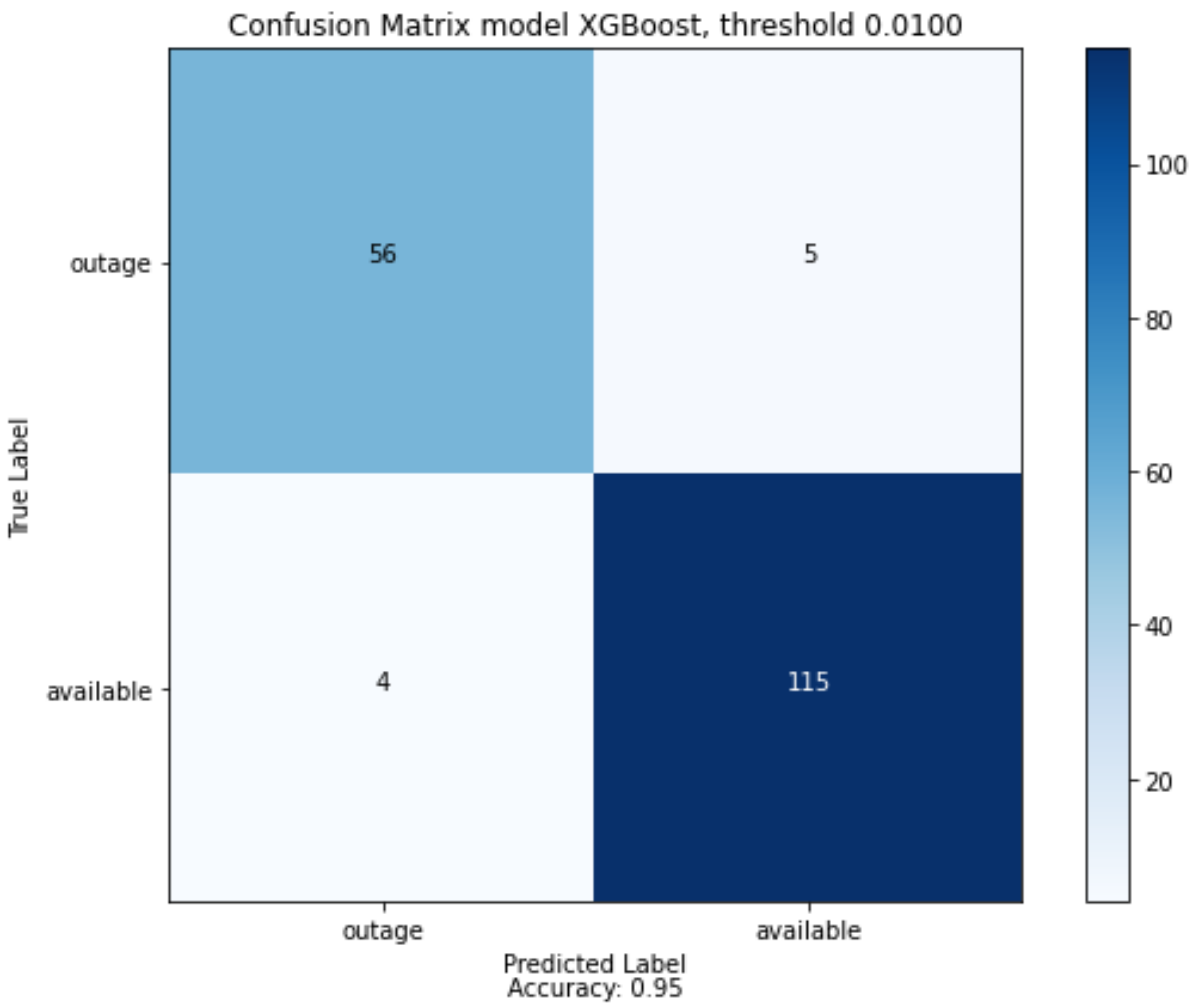

Figure 32: Confusion matrix for URLLC service-related threshold case

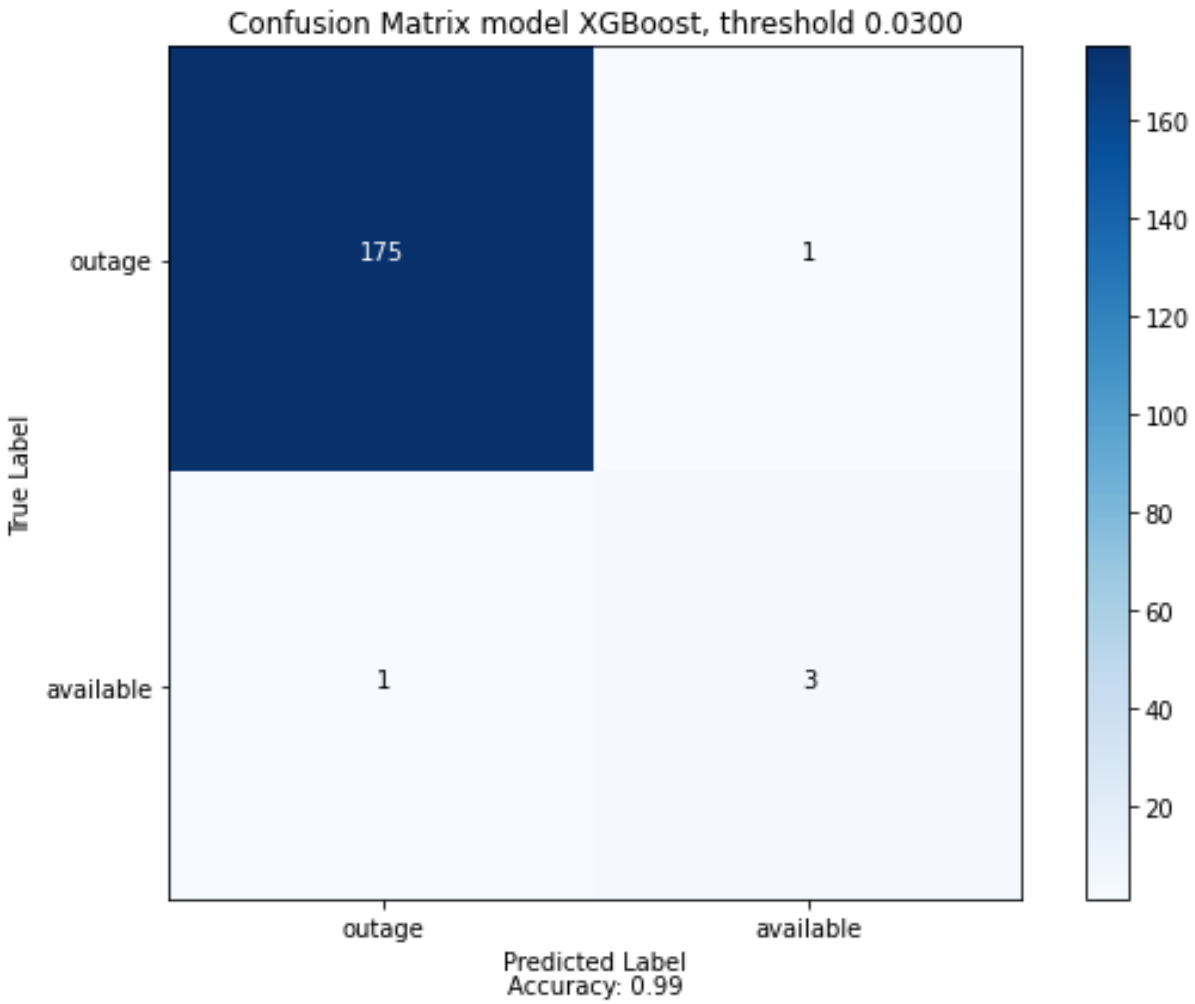

Figure 33: Confusion matrix for MTC service-related threshold case

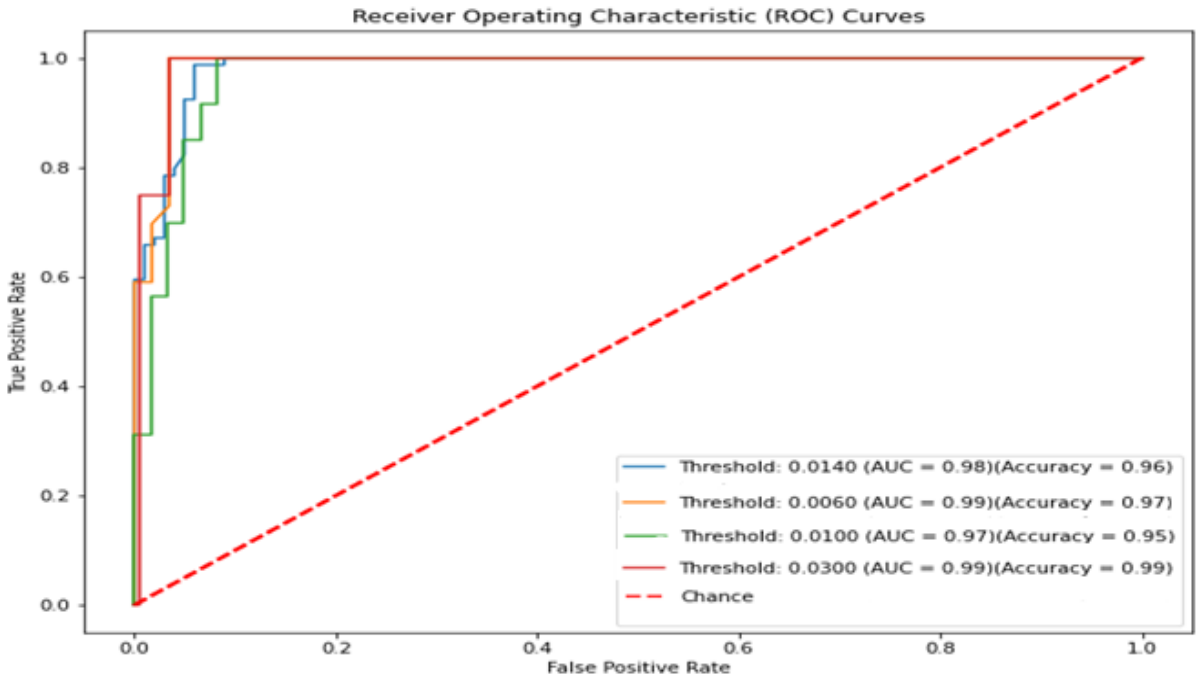

Figure 34: ROC curves for all four thresholds cases

Let it be noted here that the receiver operating characteristic curves (ROC Curves) plot (Figure 34) has the true positive rate (TPR or recall) versus the false positive rate for each of the thresholds considered. The diagonal curve represents a random classifier not being able to distinguish between the outage and availability classes, and the aim is to train a model that is represented above this diagonal. The area under the curve (AUC) indicates how many correct positive classifications can be achieved. The ROC curve is a probability curve and the AUC measures the separability, that is, the ability of the model to distinguish between the two categories. The higher the AUC, the better the model performs in classification.

Of particular importance it is to declare that the metrics that are taken into consideration are the total number of them. This is the final criterion for the selection of the optimum model, which is the quantile threshold-related model. Of particular interest and merit is F1-score, as the harmonic mean of precision and recall. This is due to the fact that both false positives and false negatives bear a special significance for the Outage Probability Prediction Model. The quantile model that we've finally picked did excel overall and particularly when it comes to these particular metrics.

6.1.3. Latency Prediction Model

To validate the ARIMA model under more realistic conditions, we applied it to the dataset generated via the Hyperledger Caliper framework, described in Section 5.2. This dataset captures the latency of blockchain operations for four consensus mechanisms RAFT, QBFT, IBFT, and SBFT, under simulated transaction loads.

Using a 60-point input window (as established in our mock data experiments), we forecasted latency values and compared them against actual measurements for each protocol. The following plots illustrate the alignment between predicted and observed values.

- **IBFT and QBFT:** Both consensus algorithms exhibit a consistent upward latency trend with increasing transaction load (Figure 35 & Figure 36). The ARIMA model demonstrates excellent fit, with MAPE values of 0.011 for IBFT and 0.007 for QBFT. These low errors confirm that ARIMA effectively captures their structured, gradually increasing latency behavior.
- **RAFT and SBFT:** The ARIMA model maintains solid performance on these protocols, showing a good fit to the overall latency behavior across the majority of the time series (Figure 37 & Figure 38). Notably, the initial segments for both RAFT and SBFT exhibit abrupt spikes in latency. These anomalies are due to system startup dynamics, events such as leader election, consensus initialization, and synchronization overhead, which occur only at the beginning of the measurement process. While the outliers could be removed from the dataset to reduce noise and improve MAPE, we chose to retain them to demonstrate the model's robustness. Despite their presence, the model quickly adapts and aligns with the stabilized behavior, achieving respectable MAPE values of 0.082 for RAFT and 0.080 for SBFT.

Table 11 reports the MAPE for each blockchain protocol.

Table 11: MAPE per Consensus Protocol

Consensus Protocol	MAPE
IBFT	0.011
QBFT	0.007
RAFT	0.082
SBFT	0.080

These results confirm that the ARIMA model, is able to generalize across diverse blockchain latency behaviors when provided with a modest historical context. Despite the noise and variability in real workloads, the model remains robust.

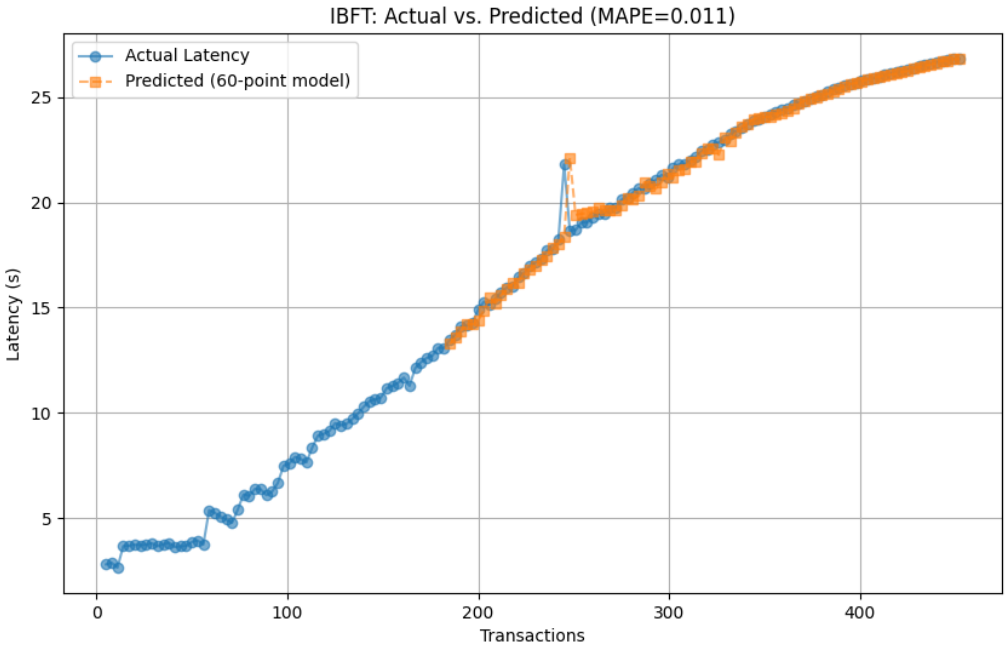

Figure 35: Real vs Predicted latency – IBFT

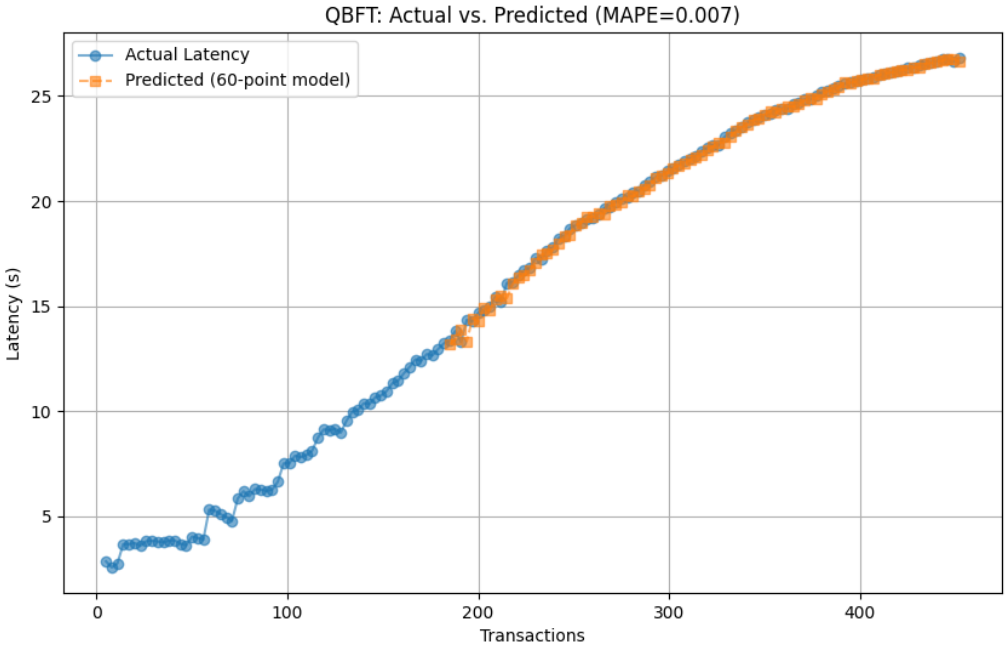

Figure 36: Real vs Predicted latency – QBFT

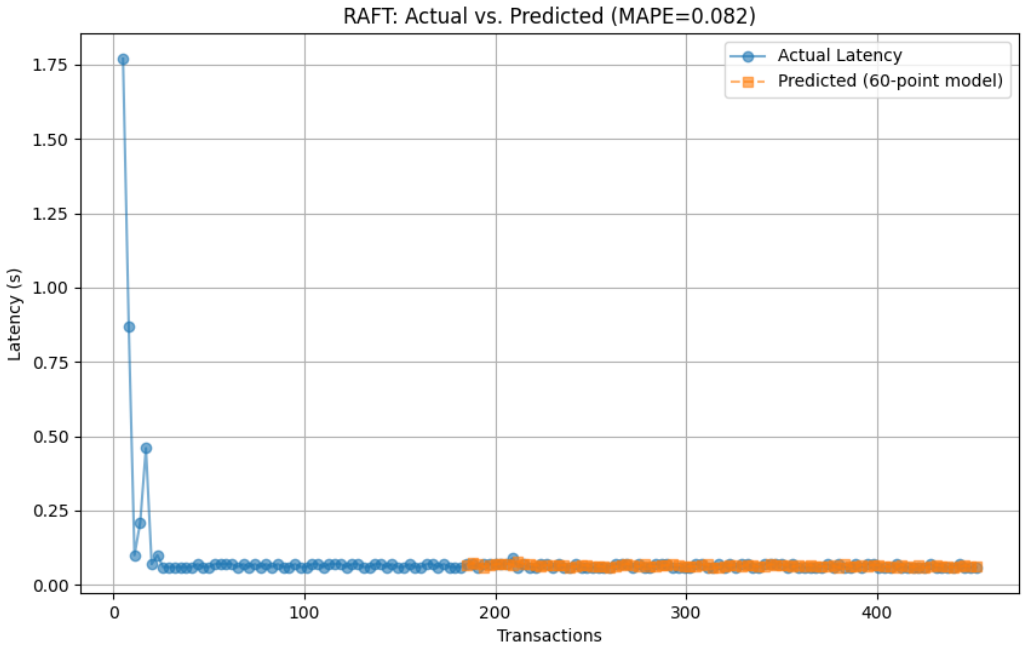

Figure 37: Real vs Predicted latency – RAFT

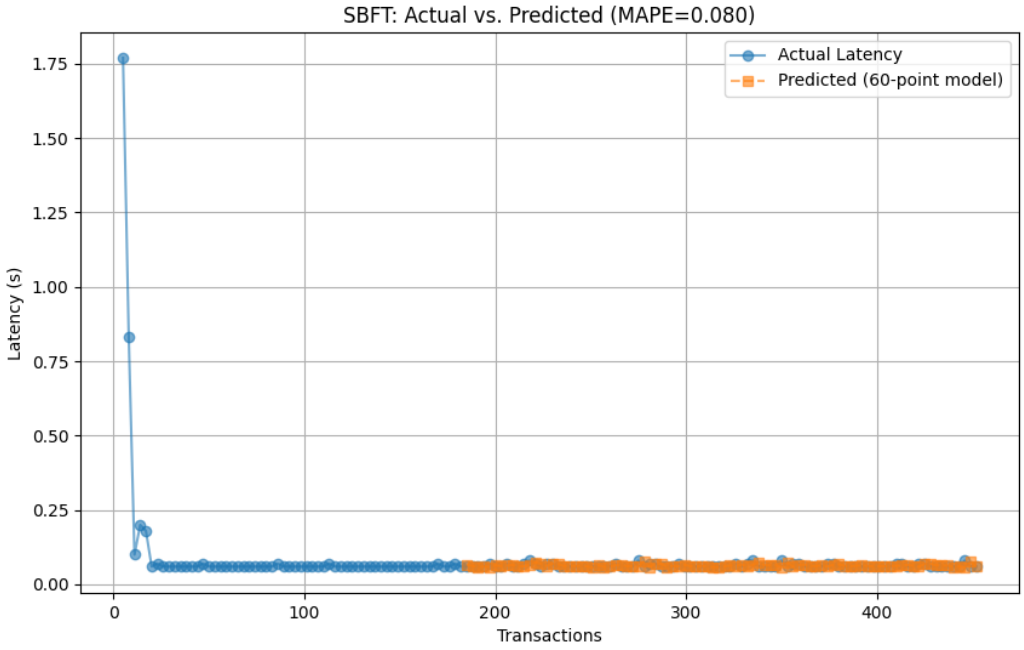

Figure 38: Real vs Predicted latency – SBFT

6.2. Complexity vs. Security Trade-offs in Consensus Protocols

Evaluating consensus protocols involves more than comparing metrics like latency or throughput; it requires a clear understanding of the assumptions they make, the environments they are designed for, and the nature of the faults they are built to tolerate [39]. In this section, we contrast several consensus mechanisms RAFT, QBFT, IBFT, SBFT and FastBFT [40], across the axes of communication complexity,

fault tolerance, and suitability for real-world deployment. This analysis builds upon the overview of section 5.1.1, providing a more detailed comparison.

RAFT: Simplicity for Trusted Environments

RAFT is a CFT protocol, meaning it assumes that faulty nodes simply stop responding rather than behaving maliciously. This makes RAFT ideal for environments where nodes are managed centrally or are otherwise trusted to behave correctly. The protocol is straightforward, leader-based, and involves only $O(n)$ message complexity, as the leader communicates directly with all followers and awaits acknowledgments from a majority.

The benefit of RAFT lies in its simplicity and performance. Because it does not attempt to protect against arbitrary or Byzantine behavior, it avoids the overhead associated with complex verification or quorum intersection logic. This makes it highly scalable and low-latency in practice.

However, this same simplicity is its primary limitation. RAFT cannot function correctly if nodes behave maliciously for example, sending conflicting messages or selectively withholding data. In adversarial settings or consortium blockchains where participants may not fully trust one another, RAFT's guarantees are insufficient.

QBFT and IBFT: Strong Guarantees at a Cost

Both QBFT and IBFT are designed to tolerate Byzantine faults, where nodes can behave arbitrarily, including maliciously. They achieve this by relying on multiple rounds of all-to-all communication to establish consensus through majority agreement among a supermajority of nodes (typically approximately 2/3).

The major strength of these protocols lies in their robustness, as they can continue to operate securely even when some nodes attempt to subvert the system. This is essential in decentralized settings where full trust among participants cannot be assumed.

The cost of this robustness, however, is their quadratic message complexity: $O(n^2)$. As the number of participants grows, so does the number of messages that must be sent and validated, leading to increasing latency and decreased throughput. These limitations have historically constrained their scalability, making them best suited for smaller permissioned networks or environments where security is prioritized over raw performance.

SBFT: Balancing Robustness and Efficiency

SBFT was introduced to address the scalability issues of traditional BFT protocols while maintaining their strong fault tolerance properties. It uses techniques like threshold cryptography and collector nodes to reduce the communication complexity to $O(n)$, even under Byzantine conditions.

This innovation allows SBFT to scale better than classical BFT protocols while still withstanding up to $\lfloor (n-1)/3 \rfloor$ Byzantine faults. In practical terms, SBFT can deliver higher throughput and lower latency than QBFT or IBFT in larger networks, without compromising on security.

Despite its advantages, SBFT remains relatively complex to implement and tune, especially due to its reliance on advanced cryptographic techniques. Moreover, while its message complexity is improved, the computational cost of threshold signature operations may become a bottleneck in resource-constrained environments.

FastBFT: Hardware-Accelerated Consensus for High Performance

FastBFT, described in detail in NANCY D5.2 'NANCY Security and Privacy Distributed Blockchain-based Mechanisms', represents a newer class of Byzantine fault-tolerant protocols that combine cryptographic message aggregation with trusted hardware. Unlike traditional approaches that rely heavily on cryptographic signatures for quorum verification, FastBFT uses Trusted Execution Environments (TEEs) to perform lightweight secret sharing and aggregation operations in a secure enclave.

FastBFT achieves linear message complexity $O(n)$, while preserving Byzantine fault tolerance. This makes it particularly attractive for high-throughput blockchain systems where both performance and security are essential. FastBFT further organizes its nodes in a tree topology to minimize bottlenecks during aggregation phases and employs an optimistic execution model in the common case, falling back to a classical BFT mode only under failure.

The inclusion of hardware-assisted security is both its strength and its caveat. FastBFT can deliver performance comparable to CFT protocols like RAFT, but it assumes the presence of trustworthy TEEs and depends on their correct functioning.

Summary of Trade-offs

Each protocol embodies a different set of trade-offs between scalability, fault tolerance, and implementation complexity:

- RAFT is optimal when simplicity and CFT are sufficient, but breaks down in the face of malicious behavior.
- QBFT/IBFT provide robust Byzantine protection but suffer from scalability limits due to their quadratic communication patterns.
- SBFT introduces optimizations that reduce message complexity, making Byzantine protection more viable at scale, though with higher computational overhead.
- FastBFT achieves the most favorable balance by leveraging hardware to offload cryptographic costs, delivering low latency and high throughput under Byzantine assumptions, but at the cost of relying on secure hardware infrastructure.

6.3. Evaluation via theory: Point Processes

6.3.1. Theoretical background: Poison Point Process

The Poisson Point Process (PPP)-based model [41] is a widely used theoretical framework in wireless network analysis. It stems from stochastic geometry, a field of applied mathematics that models spatial configurations of random objects. In the context of wireless communication, PPP is used to model the spatial distribution of BSs, assuming they are deployed at random positions over an infinite plane [42] [43].

The PPP-based model offers a powerful compromise, compared to real-world conditions:

- Tractability: It leads to elegant, often closed-form expressions for critical network performance metrics.
- Averaged Performance: It provides statistical averages over many possible network topologies, useful for macro-level planning.
- Baseline Comparisons: PPP models serve as a theoretical benchmark to compare against practical deployments or simulations.

In a PPP-based cellular network model:

- BSs are distributed according to a homogeneous PPP with density λ (units: BSs/m).
- Users are also randomly located and connect to their nearest BS.
- Signal propagation follows a path loss law, where $r^{-\alpha}$ is the distance and r the path loss exponent.
- Interference comes from all other BSs, modelled as a Poisson shot noise process.
- Rayleigh fading [44]
- is typically assumed, meaning received power follows an exponential distribution.

Given this setup, the coverage probability $P_{cov}(\theta)$ is defined as:

$$P_{cov}(\theta) = P(SINR > \theta)$$

Where θ is the Signal to Interference and Noise Ratio (SINR) threshold.

The coverage probability under PPP with Rayleigh fading and path loss exponent α is given by:

$$P_{cov}(\theta) = \int_0^{\infty} e^{-\frac{\theta\sigma^2 r^{\alpha}}{P} - \pi\lambda r^2 \beta(\theta,\alpha)} 2\pi\lambda r \, dr$$

Where:

- λ : density of BSs
- P : transmit power
- σ^2 : noise power
- $\beta(\theta,\alpha) = \theta^{\frac{2}{\alpha}} \int_{\theta^{-\frac{2}{\alpha}}}^{\infty} \frac{1}{1+u^{\frac{\alpha}{2}}} \, du$

This integral sometimes simplifies further in special cases (specific path loss exponent α).

In this task, we used the PPP model to derive a theoretical coverage probability given our real-world parameters (density, power, noise, threshold). It will serve as a ground truth benchmark to compare against predictions from ML models and field measurements.

To this end, we computed the following parameters (Table 12) based on the setup in the UMU campus dataset (please refer to section 3.2.2).

Table 12: PPP parameters

Parameter	Value	Source/Explanation
λ (BS density)	~1.29e-5 BS/m²	From 4 towers within the campus bounding box area (≈310,568 m²)
P (Transmit power)	≈1.29e-5 BS/m²	Provided network specification
σ^2(Noise power)	≈ 8.004e-14 W	Computed from thermal noise formula: κTB with 20 MHz bandwidth
α (Path loss exponent)	3.0	Standard for urban environments [45]
θ (SINR threshold)	1.0 (linear scale, 0 dB)	Chosen as a typical minimum viable SINR for coverage [46]

6.3.2. Pointwise Coverage per-UE

While the PPP model gives a statistical average over a large region, real-world measurements and predictions often require location-specific predictions. Thus, we moved from a global integral to a per-user analytical approximation that computes the coverage probability for each UE individually based on its distance to nearby towers.

For each UE, a typical way of achieving this is using a conditional SINR coverage probability under Rayleigh fading [47], where the distance to the serving BSs and the distances to interfering towers are known or approximated.

The expression used is:

$$P_{cov} = e^{-\frac{\theta\sigma^2 r^\alpha}{P}} \cdot \prod_{i=1}^{N} \frac{1}{1+\theta\left(\frac{r}{d_i}\right)^\alpha}$$

Where:

- r : Distance to serving BS
- d_i: Distance to each interfering BS
- θ : SINR threshold
- P : Transmit power
- σ^2: Noise power
- α : Path loss exponent

This equation assumes Rayleigh fading and independent interference from other towers.

These values (Table 12) plus the distance to each tower were substituted into the formula and yielded an average $P_{cov} \approx 36\%$ per UE. Compared to the global PPP integral, this per-UE method offers a more fine-grained and tailored view of performance at specific user locations. It remains theoretically grounded through the use of Rayleigh fading assumptions but incorporates actual or estimated distances to interfering towers, making it adaptable to real-world deployment geometry. However, even with this added realism, the resulting theoretical coverage probabilities were still significantly below reality. In practice, all UE instances experienced successful coverage, often with high SINR values. This discrepancy is largely due to the conservative assumptions of Rayleigh fading and the model's inability to account for practical interference mitigation techniques like scheduling, beamforming, and network optimization. Thus, while theoretically valuable, per-UE analytical coverage remains an approximation and should be interpreted with caution in high-quality deployment scenarios.

6.3.3. Improving per-UE coverage estimation: Rician Fading Model

To improve upon the limitations inherent in Rayleigh fading assumptions, we extended our analysis using the Rician fading model [48]. Unlike Rayleigh, which assumes purely non-line-of-sight conditions and models the signal amplitude as an exponentially distributed variable, Rician fading introduces a deterministic line-of-sight (LoS) component in addition to the random scattered paths. This makes it more suitable for real-world urban deployments where dominant paths from base stations to users often exist.

The move to Rician fading directly addresses the underestimation of coverage observed in the Rayleigh-based per-UE computations. Since Rayleigh assumes the worst-case signal variability, it is

overly pessimistic in environments where the propagation conditions are more stable. By incorporating Rician fading, our theoretical model more accurately captured the higher SINR values observed in the dataset.

Mathematically, if h is the channel gain, the Rician fading distribution describes the amplitude $|h|$ of the received signal as a combination of a constant LoS component and a circularly symmetric complex Gaussian (Rayleigh) component. The probability density function (PDF) of the Rician-distributed amplitude is:

$$|h| \sim Rice(K)$$

With PDF:

$$f_{|h|}(x) = \frac{x}{\sigma^2} e^{\frac{x^2+s^2}{2\sigma^2}} I_0 \frac{xs}{\sigma^2}$$

Where:

- s : Amplitude of the deterministic LoS component
- σ^2: Noise power
- $K = \frac{s^2}{2\sigma^2}$: is the Rician K-factor, representing the ratio of the power in the LoS component to the power in the scattered components
- $I_0(z) = \sum_{k=0}^{\infty} \frac{\left(\frac{1}{4}z^2\right)^k}{(k!)^2}$

A higher K-factor implies a stronger LoS presence, and when $K \to 0$, the distribution reduces to Rayleigh. In contrast, as $K \to \infty$, the distribution becomes more deterministic, approaching an AWGN-like behavior.

When computing coverage probability under Rician fading, the key mathematical tool is the moment generating function (MGF) of the fading distribution. For a given path loss and SINR threshold θ , and assuming a path loss of $r^{-\alpha}$, the Laplace transform of the interference term under Rician fading includes [49]:

$$M_h(s) = \frac{(1+K)e^{-K}}{1+s} e^{\frac{K}{1+s}}$$

Where $s = \frac{\theta r^2 \sigma^2}{P}$. This expression replaces the simpler exponential decay e^{-s} used in Rayleigh-based analysis, reflecting the more stable received power due to the LoS path.

This MGF is then integrated similarly to the PPP-based expressions, or applied in conditional SINR probability formulas, to compute per-UE coverage probability. The result is a model that more faithfully represents environments with semi-predictable signal behavior, like urban streets, campuses, or open areas with tower LoS.

By incorporating Rician fading, our theoretical model more accurately captured the higher SINR values observed in the dataset. Consequently, the coverage probabilities predicted using Rician fading will align much more closely with reality.

6.3.4. Model vs Theory vs Reality

In this section, we perform a direct comparison between our trained machine learning-based coverage prediction model, the theoretical predictions from the Rician fading model, and the actual coverage

conditions recorded in UMU dataset. Our goal is to evaluate how well theory aligns with reality, and how effectively our model bridges that gap.

For this analysis, we rely on the per-UE, distance-based coverage predictions calculated using the Rician fading model. This theoretical framework, as described in the previous section, is based on point process theory but incorporates a deterministic LoS component, making it more reflective of real-world urban deployments.

To determine whether a UE was in coverage or not according to the real measurements, we applied a simple thresholding rule to the recorded RSRQ values. Specifically, any user with a recorded real_rsrq greater than -15 dB [50] was considered to be in coverage. This binary labelling provided the ground truth for comparison. This binary labelling provided the ground truth for comparison.

We then compared this empirical coverage outcome to two other indicators:

1. The predicted coverage label from our machine learning model.
2. The theoretical coverage probability is computed using the Rician fading model.

By aligning these three sources of coverage information, we were able to quantitatively assess the accuracy of both our predictive model and the theoretical approximation, relative to the actual behavior of the 5G network in the field.

The results of our comparison are summarized in Table 13. We measured the accuracy of the model predictions and theoretical Rician fading-based estimates against the real-world coverage outcomes, defined via the RSRQ threshold. The machine learning model achieved an accuracy of 98%, while the Rician fading model achieved an accuracy of approximately 92.3%. This performance confirms that the ML-based approach aligns better with real-world measurements than the theoretical mathematical model based on point process theory.

Table 13: Predicted vs Theoretical vs Real Coverage

Comparison	Accuracy
Predicted Coverage vs Real Measurement	98.0%
Theoretical vs Real Measurement	92.3%

7. Conclusion

This deliverable presented a detailed and practical NIF that helps the users study and improve B-RANs in different types of real-world setups. Using smart AI models, the system makes it easier to look at how networks perform and make better choices based on real data.

Three AI prediction models were developed that focus on three main network metrics: coverage, outage, and Blockchain-induced latency. Each model was trained and tested using either open data or the NANCY testbeds, making sure they work well in practice and not just in theory.

The **Coverage Prediction Model** was trained using transfer learning with data from the University of Murcia's NANCY testbed. It showed good performance by predicting RSRQ values with a low error. More specifically, a MAE of 0.137 and a MSE of 0.218 were achieved. Furthermore, it has very low execution time, as it takes less than 1 millisecond per prediction. This makes it highly suitable for real-time network assessments.

The **Outage Prediction Model** used data from the Colosseum dataset. Different threshold levels were tested, and it was found that the quantile threshold gives the best results, especially on the F1-score. This is a metric of major importance, given the need to minimize both false positives and false negatives in outage detection. The high ROC-AUC values across configurations underscore the reliability of this model in identifying network disruptions under varying service quality requirements.

The **Latency Prediction Model**, grounded in ARIMA time series forecasting, addresses a unique challenge: assessing the latency impact of diverse blockchain consensus protocols. Applied to transaction data generated via Hyperledger Caliper, the model successfully generalized across IBFT, QBFT, RAFT, and SBFT protocols, with MAPEs as low as 0.07 for QBFT and 0.011 for IBFT. We also explored the trade-off between security and latency in consensus protocols, highlighting the performance implications of various design choices.

Finally, we evaluated the coverage prediction model using theoretical point process modelling, specifically incorporating both Rayleigh and Rician fading assumptions. This allowed us to establish a principled baseline grounded in wireless communication and point process theory. By comparing these theoretical estimates to the machine learning model's predictions and to real-world measurement data, we were able to assess the fidelity of each approach. The trained model demonstrated alignment with the measured data, validating its effectiveness in practical deployments.

All these features come together in an easy-to-use **web platform** where users can upload data, run tests, and see network performance results. It's not just a tool, but a helpful system for better planning and decision-making in B-RANs.

Bibliography

[1] L. Bonati, S. D'oro, M. Polese, S. Basagni and T. Melodia, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," IEEE Communications Magazine, vol. 59, no. 10, Oct. 2021.

[2] X. Ling, Y. Le, J. Wang, Z. Ding and X. Gao, "Practical Modeling and Analysis of Blockchain Radio Access Network," IEEE Transactions on Communications, vol. 69, no. 2, pp. 1021-1037, Feb. 2021.

[3] C. Bento, "Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis," 21 Sep. 2021. [Online]. Available: https://medium.com/data-science/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

[4] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, “Beyond throughput, the next generation,” 11th ACM Multimedia Systems Conference, 2020, pp. 303–308.

[5] “4G LTE Speed Dataset and Bandwidth.” [Online]. Available: https://www.kaggle.com/datasets/aeryss/lte-dataset/data

[6] J. Gallego-Madrid, L. Bernal-Escobedo, R. Asensio, A. Hermosilla, A. Zarca, J. Ortiz, R. Sanchez-Iborra and A. Skarmeta, "GAIA 5G: A Multi-access Smart-Campus Architecture," Lecture Notes in Computer Science, Springer International Publishing, 2022, pp. 363-374.

[7] R. Asensio-Garriga, A. Pogo Medina, G. Alarcon-Hellin, L. Escobedo, R. Sanchez-Iborraand and A. Skarmeta Gómez, "University of Murcia 5G Dataset 1," May 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15516876.

[8] “QMI-Supervisor - a 5G QMI modem controller," 2024. [Online]. Available: https://github.com/ANTS-research-group/qmi-supervisor

[9] Z. Mo, W. Su, S. Batalama and J. Matyjas, "Cooperative Communication Protocol Designs Based on Optimum Power and Time Allocation," IEEE Transactions on Wireless Communications, vol. 13, pp. 4283-4296, Aug. 2014.

[10] Z. Shi, H. Wang, Y. Fu, G. Yang, S. Ma and X. Ye, "Outage performance and optimal design of MIMO-NOMA enhanced small cell networks with imperfect channel-state information," China Communications, vol. 18, pp. 107-128, Oct. 2021.

[11] A. Traßl, E. Schmitt, T. Hößler, L. Scheuvens, N. Franchi, N. Schwarzenberg and G. Fettweis, "Outage prediction for ultra-reliable low-latency communications in fast fading channels," EURASIP Journal on Wireless Communications and Networking, vol. 2021, p. 1–25, Apr. 2021.

[12] V. Mordachev and S. Loyka, "On node density - outage probability tradeoff in wireless networks," IEEE International Symposium on Information Theory, 2008, pp. 191-195.

[13] C. Kourogiorgas, A. Panagopoulos and J. Kanellopoulos, "A New Method for the Prediction of Outage Probability of LOS Terrestrial Links Operating Above 10 GHz," IEEE Antennas and Wireless Propagation Letters, vol. 12, pp. 516-519, Apr. 2013.

[14] T. Routtenberg and J. Tabrikian, "A General Class of Outage Error Probability Lower Bounds in Bayesian Parameter Estimation," IEEE Transactions on Signal Processing, vol. 60, pp. 2152-2166, May 2012.

[15] M. Alias, N. Saxena and A. Roy, "Efficient Cell Outage Detection in 5G HetNets Using Hidden Markov Model," IEEE Communications Letters, vol. 20, pp. 562-565, Mar. 2016.

[16] H. Wang, L. Xu, Y. Tao and X. Wang, "OP Performance Prediction for Complex Mobile Multiuser Networks Based on Extreme Learning Machine," IEEE Access, vol. 8, pp. 14557-14564, Jan. 2020.

[17] N. Simmons, D. Simmons and M. Yacoub, "Outage Performance and Novel Loss Function for an ML-Assisted Resource Allocation: An Exact Analytical Framework," IEEE Transactions on Machine Learning in Communications and Networking, vol. 2, pp. 335-350, Feb. 2024.

[18] IBM, "What is a decision tree?" [Online]. Available: https://www.ibm.com/think/topics/decision-trees

[19] IBM, "What is boosting?" [Online]. Available: https://www.ibm.com/topics/boosting

[20] R. Fekadu, A. Getachew, Y. Tadele, N. Ali, and I. Goytom, "Machine Learning Models Evaluation and Feature Importance Analysis on NPL Dataset," 2022, arXiv. doi: 10.48550/ARXIV.2209.09638.

[21] X. Niu, L. Wang, and X. Yang, "A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised," 2019, arXiv. doi: 10.48550/ARXIV.1904.10604

[22] M. Saleh and N. Abd-Alsabour, "Improved Decision Tree, Random Forest, and XGBoost Algorithms for Predicting Client Churn in the Telecommunications Industry," International Journal of Advanced Computer Science and Applications, vol. 15, p. 2024, 2024.

[23] "Colosseum O-RAN COMMAG Dataset," [Online]. Available: https://github.com/wineslab/colosseum-oran-commag-dataset

[24] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," *USENIX Annual Technical Conference (ATC'14)*, 2014, pp. 305 - 320.

[25] R. Saltini, Enterprise Ethereum Alliance. "QBFT Blockchain Consensus Protocol Specification v1", Nov. 2022. [Online]. Available: https://entethalliance.github.io/client-spec/qbft_spec.html

[26] H. Moniz, "The Istanbul BFT Consensus Algorithm," 2020, arXiv. doi: 10.48550/ARXIV.2002.03613.

[27] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, D.-A. Seredinschi, O. Tamir and A. Tomescu, "SBFT: a Scalable and Decentralized Trust Infrastructure," 2019, pp. 568-580.

[28] L. Lamport, R. Shostak and M. Pease, "The Byzantine Generals Problem," International ACM Transactions on Programming Languages and Systems, vol. 4, p. 382–401, July 1982.

[29] B. Gulnara, A. Ahmad, Mirzakulova and Ж. Sharafat & Ибраева, "Time Series Forecasting By The ARIMA Method," *Scientific Journal of Astana IT University,* p. 14–23, 2022.

[30] S. Siami-Namini and A. S. Namin, “Forecasting Economics and Financial Time Series: ARIMA vs. LSTM,” 2018, arXiv. doi: 10.48550/ARXIV.1803.06386.

[31] “Hyperledger Fabric.“ [Online]. Available: https://www.lfdecentralizedtrust.org/projects/fabric

[32] “The Raft Consensus Algorithm.” [Online]. Available: https://raft.github.io/

[33] Y. Manevich, Y. Tock, and H. Meir, “Hyperledger Fabric v3: Delivering Smart Byzantine Fault Tolerant consensus,” Sep. 2024. [Online]. Available: https://www.lfdecentralizedtrust.org/blog/hyperledger-fabric-v3-delivering-smart-byzantine-fault-tolerant-consensus.

[34] “Besu,” [Online]. Available: https://www.lfdecentralizedtrust.org/projects/besu .

[35] “Configure QBFT consensus,” Apr. 2025. [Online]. Available: https://besu.hyperledger.org/private-networks/how-to/configure/consensus/qbft

[36] “Configure IBFT 2.0 consensus,” Apr 2025. [Online]. Available: https://besu.hyperledger.org/private-networks/how-to/configure/consensus/ibft

[37] “Caliper,” [Online]. Available: https://www.lfdecentralizedtrust.org/projects/caliper .

[38] “Node.js — Run JavaScript Everywhere,” [Online]. Available: https://nodejs.org/

[39] G. Zhang, F. Pan, Y. Mao, S. Tijanic, M. Dang'ana, S. Motepalli, S. Zhang and H.-A. Jacobsen, “Reaching Consensus in the Byzantine Empire: A Comprehensive Review of BFT Consensus Algorithms,” 2022, arXiv. doi: 10.48550/ARXIV.2204.03181

[40] J. Liu, W. Li, G. O. Karame and N. Asokan, "Scalable Byzantine Consensus via Hardware-assisted Secret Sharing," IEEE Transactions on Computers, vol. 68, p. 139–151, Jan. 2016.

[41] S. N. Chiu, D. Stoyan, W. S. Kendall and J. Mecke, Stochastic Geometry and Its Applications, Wiley, 2013.

[42] F. Baccelli and B. Blaszczyszyn, Stochastic Geometry and Wireless Networks: Volume I Theory, 2010.

[43] M. Haenggi, J. G. Andrews, F. Baccelli, O. Dousse and M. Franceschetti, "Stochastic geometry and random graphs for the analysis and design of wireless networks," IEEE Journal on Selected Areas in Communications, vol. 27, no. 7, pp. 1029-1046, Aug. 2009.

[44] B. Sklar, "Rayleigh fading channels in mobile digital communication systems. I. Characterization," IEEE Communications Magazine, vol. 35, pp. 136-146, Sep. 1997.

[45] A. Imoize and A. Oseni, "Investigation and pathloss modeling of fourth generation long term evolution network along major highways in Lagos Nigeria," Ife Journal of Science, vol. 21, p. 39, Apr. 2019.

[46] "SINR - Teltonika Networks Wiki" [Online]. Available: https://wiki.teltonika-networks.com/view/SINR

[47] H. P. Keeler, B. Blaszczyszyn and M. K. Karray, "SINR-based k-coverage probability in cellular networks with arbitrary shadowing," IEEE International Symposium on Information Theory, 2013, pp. 1167-1171.

[48] A. Abdi, C. Tepedelenlioglu, M. Kaveh and G. Giannakis, "On the estimation of the K parameter for the Rice fading distribution," IEEE Communications Letters, vol. 5, no. 3, pp. 92-94, Mar. 2001.

[49] S. O. Rice, "Statistical properties of a sine wave plus random noise," The Bell System Technical Journal, vol. 27, pp. 109-157, 1948.

[50] "RSRP and RSRQ - Teltonika Networks Wiki," [Online]. Available: https://wiki.teltonika-networks.com/view/RSRP_and_RSRQ