



Dissertation on

**Machine Intelligence based Quality of Service Framework to
Address Network Slicing Issues in 5G**

Submitted in partial fulfillment of the requirements for the award of degree of

**Bachelor of Technology
in
Computer Science & Engineering**

UE20CS461A – Capstone Project Phase - 2

Submitted by:

A Mohan Kumar	PES2UG20CS002
Allu Vineela Sravanthi	PES2UG20CS036
Sathyanarayana RK	PES2UG20CS315
Suhas T J	PES2UG20CS355

Under the guidance of

Dr. Vinodha K
Associate Professor
PES University

June - Nov 2023

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

Machine Intelligence Based Quality of Service Framework to Address Network Slicing Issues in 5G

is a bonafide work carried out by

A Mohan Kumar

PES2UG20CS002

Allu Vineela Sravanthi

PES2UG20CS036

Sathyanarayana RK

PES2UG20CS315

Suhas T J

PES2UG20CS355

In partial fulfillment for the completion of seventh semester Capstone Project Phase - 2 (UE20CS461A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June 2023 – Nov. 2023. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature

Dr. Vinodha K

Associate Professor

Signature

Dr. Sandesh B J

Chairperson

Signature

Dr. B K Keshavan

Dean of Faculty

Name of the Examiners

Signature with Date

1. _____

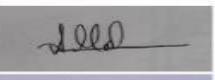
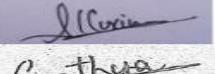
2. _____

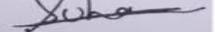
DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled "**Machine Intelligence Based Quality Of Service Framework To Address Network Slicing Issues in 5G**" has been carried out by us under the guidance of Dr.Vinodha K, Associate Professor and submitted in partial fulfillment of the course requirements for the award of degree of **Bachelor of Technology in Computer Science and Engineering** of PES University, Bengaluru during the academic semester June – Nov. 2023. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

A Mohan Kumar

PES2UG20CS002



Allu Vineela Sravanthi

PES2UG20CS036

Sathyanarayana RK

PES2UG20CS315

Suhas T J

PES2UG20CS355

ACKNOWLEDGEMENT

We would like to express our gratitude to **Dr. Vinodha K**, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE20CS461A -Capstone Project Phase – 2.

We are grateful to the Capstone Project Coordinator, Dr. Sarasvathi V, Professor and **Dr. Sudeepa Roy Dey**, Associate Professor, for organizing, managing, and helping with the entire process.

We take this opportunity to thank **Dr. Sandesh B J**, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support we have received from the department. We would like to thank **Dr. B.K. Keshavan**, Dean of Faculty, and PES University for his help.

We are deeply grateful to **Dr. M. R. Doreswamy**, Chancellor, PES University, **Prof. Jawahar Doreswamy**, Pro Chancellor – PES University, **Dr. Suryaprasad J**, Vice-Chancellor, PES University and **Prof. Nagarjuna Sadineni**, Pro-Vice Chancellor – PES University, for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement we have received from my family and friends.

ABSTRACT

In the rapidly advancing realm of 5G networks, the proliferation of 5G technologies has introduced a critical feature known as network slicing, allowing the creation of specialized virtual networks tailored to diverse applications. However, the challenge lies in ensuring Quality of Service (QoS) metrics for each slice in dynamic scenarios. To address this, a comprehensive analysis was conducted, exploring traditional algorithms, plain supervised models, and deep learning techniques using the SimPy simulation tool to construct a basic architecture for a traditional approach. Traditional algorithms and supervised models serve as benchmarks, highlighting their constraints in the dynamic 5G landscape. The study aims to compare traditional and machine intelligence approaches, focusing on determining which method offers the most accurate resource allocation to the slices. This research endeavors to provide insights into the efficiency of different approaches in addressing the intricate challenges of resource allocation in the context of 5G network slicing, contributing to the advancement of this transformative technology.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1	INTRODUCTION	1 - 2
2	PROBLEM STATEMENT	3
3	LITERATURE REVIEW 3.1 Resource Allocation in Network Slicing: Leveraging Deep Reinforcement Learning 3.2 Synergetic Resource Allocation in Network Slicing: Enhanced Performance Through Distributed Bandwidth and Computational Allocation 3.3 Revolutionizing Networking Landscapes: Unveiling the Potential of Network Slicing in Customized Service Delivery and Resource Management 3.4 Real-time Resource Orchestration for Tailored Network Services: Implementing Dynamic Allocation in Network Slicing 3.5 Optimizing End-to-End Network Slicing: Multi-Agent Reinforcement Learning for Resource Allocation and Adaptability 3.6 Secure5G: Deep Learning-Powered Framework for Safeguarding Network Slicing Security 3.7 Deep Learning and Blockchain Integration for Enhanced Security in 5G-Enabled IoT Networks 3.8 Machine Learning-Driven Automation for 5G Network Slice Control Functions 3.9 Enhancing 5G Radio Access Network Slicing Through Intelligent Resource Scheduling 3.10 Exploring the Landscape of 5G Network Slicing: Challenges, Opportunities, and Use Cases	4 - 17 4 - 5 5 - 6 6 - 7 8 - 9 10 - 11 11 - 12 12 - 13 14 - 15 15 - 16 16 - 17

4	PROJECT REQUIREMENTS SPECIFICATIONS	18 - 25
	4.1 System Features	18 - 19
	4.2 Operating Environment	19 - 20
	4.3 General Constraints, Assumptions and Dependencies	20 - 21
	4.4 Risks	21 - 22
	4.5 Functional Requirements	22 - 23
	4.6 Hardware Requirements	23 - 24
	4.7 Software Requirements	24
	4.8 Performance Requirements	24 - 25
5	SYSTEM DESIGN	26 - 34
	5.1 Current System	26
	5.2 Design Goals	26 - 27
	5.3 Architecture Choices	27 - 28
	5.4 High Level System Design	28
	5.5 5G Network Slicing Architecture Overview	29
	5.6 Master Class Diagram	30 - 31
	5.7 Packaging and Deployment Diagram	31
	5.8 Help	31 - 32
	5.9 Design Details	32 - 33
	5.10 Resource Utilization	33 - 34
6	PROPOSED METHODOLOGY	35 - 40
	6.1 Traditional Algorithms Approach	35 - 38

	6.2 Machine Learning Algorithms Approach	38 - 40
	6.3 Project Workflow	40
7	IMPLEMENTATION AND PSEUDOCODE	41 - 71
	7.1 Traditional Algorithms Implementation	41 - 44
	7.2 Simulation	44 - 57
	7.3 Introduction to Dataset	57- 59
	7.4 Machine Learning Models Review	59 - 71
8	RESULTS AND DISCUSSION	72
9	CONCLUSION AND FUTURE WORK	73 - 74
REFERENCES/BIBLIOGRAPHY		75 -76
APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS		77
IEEE DRAFT PAPER		78 - 82
CAPSTONE PROJECT POSTER		83

LIST OF FIGURES

Sl. No.	NAME OF THE FIGURE	Page no.
1	5G Network Slicing Architecture	29
2	Details About Types of Network Slices	29
3	Master Class Diagram	30
4	Packaging and Deployment Diagram	31
5	Priority Allocation Results	37
6	Machine Learning Approach Workflow	40
7	Dynamic Resource Allocation Results	42
8	Four Different Network Scenarios	55
9	Heterogeneous network	56
10	5G Network Slicing Dataset	58
11	Random Forest Classifier	59
12	AdaBoost Classifier	60
13	Gradient Boosting Classifier	61
14	Extra Trees Classifier	62
15	Bagging Classifier	63
16	Decision Tree Classifier	64
17	Multi-Layer Perceptron	65
18	ML Models Performance results	66
19	Stacking Classifier	67
20	Voting Classifier	68

LIST OF TABLES

TABLE NO.	TITLE	Page no.
1	Traditional Algorithms	44
2	Simulation Results	56
3	Machine learning Models	72

CHAPTER 1

INTRODUCTION

1.1 Network Slicing

Network slicing is a fundamental component of 5G networks that allows service providers to create virtual networks tailored to specific use cases. It enables the deployment of multiple logical networks on a common physical infrastructure, providing dedicated bandwidth, latency, and reliability for each use case.

Network slicing facilitates the provision of differentiated services with specific QoS requirements such as enhanced mobile broadband, massive machine type communications, and ultra-reliable low latency communications. It also optimizes resource utilization by sharing the physical infrastructure across different use cases.

To achieve network slicing, service providers use various slicing techniques, including software-defined networking (SDN), network function virtualization (NFV), and cloud computing. These techniques enable the creation, management, and orchestration of virtual networks.

Overall, network slicing is a crucial feature of 5G networks that enables service providers to offer differentiated services with specific QoS requirements, while also optimizing resource utilization. It has the potential to drive innovation and create new business opportunities in various industries, including healthcare, transportation, and manufacturing.

1.2 Dynamic Resource Allocation

Dynamic resource allocation is an essential feature of 5G networks that enables service providers to allocate network resources efficiently based on current traffic patterns and user requirements. It ensures that critical use cases receive the required resources in real-time, resulting in an enhanced user experience.

Dynamic resource allocation enables service providers to allocate resources such as bandwidth, processing power, and storage capacity dynamically, optimizing the network's overall performance. To achieve this, various techniques such as machine learning, predictive

algorithms and artificial intelligence are used to predict traffic patterns and allocate resources in real-time.

Overall, dynamic resource allocation plays a crucial role in ensuring that users receive the required QoS metrics, resulting in an enhanced user experience. It is expected to drive innovation and create new business opportunities in various industries, including healthcare, transportation, and manufacturing.

1.3 Quality of Service Framework

A Quality of Service (QoS) framework is a set of guidelines and policies that govern the allocation of network resources, ensuring that users receive the required bandwidth, latency, and reliability for their specific use case. It is an essential component of 5G networks that enables service providers to prioritize traffic based on user requirements and optimize network performance.

The QoS framework provides mechanisms for monitoring and controlling network traffic to ensure that QoS metrics are met. Service providers use various techniques such as traffic shaping, packet scheduling, and admission control to allocate network resources efficiently and ensure that users receive the required QoS metrics.

Overall, a QoS framework plays a crucial role in enhancing the user experience and optimizing network performance. It is expected to drive innovation and create new business opportunities in various industries, including healthcare, transportation, and manufacturing.

CHAPTER 2

PROBLEM STATEMENT

In the dynamic realm of 5G networks, where diverse applications demand exemplified Quality of Service (QoS), efficiently allocating network resources has become an intricate challenge. While network slicing holds the promise of customized virtual networks, the effective distribution and real-time management of resources among these slices necessitate an intelligent QoS framework. Issues such as congestion, interference, and suboptimal resource allocation often lead to compromised QoS, impacting user experience negatively. Traditional methods fall short in addressing these intricacies. Our project bridges this gap through an exhaustive comparative study encompassing traditional algorithms, supervised machine learning models, and cutting-edge reinforcement learning techniques.

Our primary goal is dual-fold:

1. Firstly, to empirically establish that supervised machine learning surpasses traditional models in the context of 5G network slicing. Rigorous analysis substantiates its superior predictive accuracy and adaptability, especially in dynamic network conditions.
2. Secondly, our project endeavors to introduce a pragmatic and intelligent QoS framework. This framework not only sets a benchmark for network administrators but also equips them with real-time insights. By offering practical recommendations for optimal network slice selection tailored to specific applications, our QoS framework emerges as an indispensable tool for network administrators navigating the complexities of 5G networks. The project culminates in the development of an unyielding framework ensuring unwavering Quality of Service in 5G network slicing, addressing a critical need in modern network management.

CHAPTER 3

LITERATURE REVIEW

3.1 Resource Allocation in Network Slicing: Leveraging Deep Reinforcement Learning [1] (RESEARCH PAPER – 1)

3.1.1 Introduction:

A potential method called network slicing allows operators to divide a single physical network into numerous logical networks, each of which is optimized for a particular use case. The complex and dynamic nature of the network environment makes it difficult to manage resources effectively in network slicing. The authors of this work suggest a deep reinforcement learning (DRL) method.

3.1.2 Characteristics and Implementation:

The proposed DRL approach is based on a multi-agent system, where each agent represents a slice and is responsible for managing the resources of that slice. The agents use a deep neural network to learn the optimal resource allocation policy for their slice. The network input includes the current network state and the slice's resource demand, and the output is the resource allocation decision.

3.1.3 Features:

The DRL approach proposed in this paper has several key features. First, it is designed to handle the dynamic and complex nature of the network environment, allowing it to adapt to changes in network conditions. Second, it is able to learn from experience, enabling it to improve over time and optimize network performance. Third, it is a distributed approach that can be applied to large-scale networks.

3.1.4 Evaluation:

The authors evaluate the proposed DRL approach using a simulation environment based on a real-world network topology. The simulation results finalize that the DRL approach outperforms traditional resource management methods in terms of network performance metrics such as throughput, delay, and packet loss. The authors also perform a sensitivity analysis to evaluate the robustness of the DRL.

3.2 Synergetic Resource Allocation in Network Slicing: Enhanced Performance Through Distributed Bandwidth and Computational Allocation [5] (RESEARCH PAPER - 2.)

3.2.1 Introduction:

Network slicing is a promising technique for providing customized network services to different users or applications. However, efficient resource allocation is crucial for ensuring the quality of service (QoS) and maximizing the utilization of network resources. In this paper, the authors propose a distributed resource allocation approach for network slicing, which jointly allocates both bandwidths.

3.2.2 Characteristics and Implementation:

The proposed approach is based on a multi-agent system, where each agent represents a network slice and is responsible for managing the resources of that slice. The agents communicate with each other to coordinate resource allocation decisions and ensure that the overall network performance is optimized.

The authors propose a distributed optimization algorithm that combines both primal-dual subgradient methods and consensus-based algorithms to achieve efficient resource allocation. The proposed approach also takes into account the resource heterogeneity and dynamic network conditions, such as changing user demands.

3.2.3 Features:

The proposed approach has several key features. First, it is a distributed approach that allows for efficient and scalable resource allocation in large-scale network slicing scenarios. Second, it jointly considers both bandwidth and computational resources, which are critical for many emerging applications. Third, it takes into account the dynamic nature of the network environment and can adapt to changes in user demands and network conditions. Finally, the authors demonstrate through simulations that the proposed approach can achieve better network performance compared to traditional resource allocation methods.

3.2.4 Evaluation:

The authors evaluate the proposed approach through simulations based on a real-world network topology. The simulation results displays that the proposed approach can achieve better network performance in terms of resource utilization and QoS compared to traditional resource allocation methods. The suggested approach is also subjected to a sensitivity analysis by the authors, who demonstrate its adaptability to changing user needs and network conditions as well as its robustness under various network settings. Overall, the outcomes show how well the suggested method for distributing resource allocation in network slicing works.

3.3 Revolutionizing Networking Landscapes: Unveiling the Potential of Network Slicing in Customized Service Delivery and Resource Management [6] (RESEARCH PAPER - 3)

3.3.1 Introduction:

A promising idea in networking is network slicing, which enables the division of a single physical network infrastructure into several virtual networks, each of which is optimized for a particular set of Services or applications. This enables network operators to offer differentiated services to their customers, with each slice providing a unique set of capabilities, performance guarantees, and security features. The implementation of network slicing involves several challenges related to the design of the underlying architecture, orchestration of resources, and management of network slices.

3.3.2 Characteristics and Implementation:

Network slicing is characterized by several key features, including the ability to support multiple virtual networks on a single physical infrastructure, the ability to dynamically allocate and manage resources across network slices, and the ability to isolate network slices from each other to ensure security and privacy. In order to construct virtual networks that may be customized to the unique requirements of various applications and services, network slicing is implemented using virtualization technologies, such as Software-defined Networking and Network Function Virtualization.

3.3.3 Features:

The features of network slicing can be broadly categorized into three categories: resource isolation, service differentiation, and dynamic resource management. Resource isolation involves the ability to create isolated virtual networks that are dedicated to specific applications or services, with each network having its own set of resources and performance guarantees. Service differentiation involves the ability to provide different levels of service quality, security, and reliability to different applications or services. Dynamic resource management involves the ability to allocate and manage resources across network slices in real-time, based on the changing needs of applications and services.

3.3.4 Evaluation:

The evaluation of network slicing involves several metrics, including network efficiency,

scalability, flexibility, and security. Network efficiency is measured by the ability to maximize the use of available resources, while maintaining a high level of performance and service quality. Scalability is measured by the ability to support a large number of network slices, each with different requirements and characteristics. Flexibility is measured by the ability to adapt to changing network conditions and requirements, while maintaining a high level of service quality. Security is measured by the ability to ensure the isolation and privacy of network slices, and to prevent unauthorized access and attacks.

3.4 Real-time Resource Orchestration for Tailored Network Services: Implementing Dynamic Allocation in Network Slicing [4] (RESEARCH PAPER – 4)

3.4.1 Introduction:

Network slicing is a key enabler for 5G and beyond networks, as it allows network operators to offer differentiated services to their customers. Dynamic virtual resource allocation is a technique used in network slicing that enables resources to be allocated and de-allocated to network slices in real-time, based on the changing needs of applications and services. This paper discusses the implementation of dynamic virtual resource allocation in network slicing, and the challenges and opportunities it presents.

3.4.2 Characteristics and Implementation:

Dynamic virtual resource allocation involves the use of Software-Defined Networking and Network Function Virtualization technologies to build virtual networks that can be tailored to the relevant requirements of distinct applications and services. This enables resources to be allocated and de-allocated in real-time, based on the changing needs of applications and services. The implementation of dynamic virtual resource allocation involves the deployment of a centralized controller that manages the allocation of resources to network slices.

3.4.3 Features:

The features of dynamic virtual resource allocation include the ability to allocate resources in real-time, based on the changing needs of applications and services, the ability to optimize resource utilization across network slices, and the ability to provide different levels of service quality, security, and reliability to different applications and services. Dynamic virtual resource allocation also enables network operators to offer flexible pricing models, as resources can be allocated and de-allocated on a per-usage basis.

3.4.4 Evaluation:

The evaluation of dynamic virtual resource allocation involves several metrics, including resource utilization, service quality, scalability, and cost-effectiveness. Resource utilization is measured by the ability to maximize the use of available resources, while maintaining a high level of performance and service quality. Service quality is measured by the ability to provide different levels of service quality, security, and reliability to different applications and services. Scalability is measured by the ability to support a large number of network slices, each with different requirements and characteristics. Cost-effectiveness is measured by the ability to optimize resource utilization and minimize costs, while maintaining a high level of service quality.

3.5 Optimizing End-to-End Network Slicing: Multi-Agent Reinforcement Learning for Resource Allocation and Adaptability [2](RESEARCH PAPER - 5)

3.5.1 Introduction:

End-to-end network slicing is a key feature of 5G and beyond networks that enables network operators to offer differentiated services to their customers. Resource management is a crucial aspect of end-to-end network slicing, as it involves the allocation of resources to network slices to ensure optimal performance and service quality. This paper proposes a multi-agent reinforcement learning based approach to resource management in end-to-end network slicing, and discusses the challenges and opportunities it presents.

3.5.2 Characteristics and Implementation:

The proposed approach involves the use of multi-agent reinforcement learning (MARL) to manage resources in end-to-end network slicing. MARL involves the use of multiple agents, each responsible for managing a different aspect of the network slice, to optimize resource allocation and performance. The implementation of the approach involves the deployment of a centralized controller that manages the agents and their interactions, and the use of a reward function to incentivize agents to optimize resource allocation and performance.

3.5.3 Features:

The features of the proposed MARL-based approach include the ability to optimize resource allocation and performance in end-to-end network slicing, the ability to adapt to changing network conditions and requirements, and the ability to provide different levels of service quality, security, and reliability to different applications and services. The approach also enables network operators to offer flexible pricing models, as resources can be allocated and de-allocated on a per-usage basis.

3.5.4 Evaluation:

The evaluation of the MARL-based approach involves several metrics; including resource utilization, service quality, scalability, and convergence time. Resource utilization is measured by the ability to maximize the use of available resources, while maintaining a high level of performance and service quality. Service quality is measured by the ability to provide different levels of service quality, security, and reliability to different applications and services. Scalability is measured by the ability to support a large number of network slices, each with different requirements and characteristics. Convergence time is measured by the time required for the agents to converge to an optimal resource allocation and performance.

3.6 Secure5G: Deep Learning-Powered Framework for Safeguarding Network Slicing Security [7]

(RESEARCH PAPER – 6)

3.6.1 Introduction:

Network slicing is a key feature of 5G and beyond networks that enables network operators to offer differentiated services to their customers. However, network slicing also introduces security challenges, as different network slices may have different security requirements and vulnerabilities. This paper proposes a deep learning-based framework, called Secure5G, for secure network slicing in 5G and beyond networks, and discusses the challenges and opportunities it presents.

3.6.2 Characteristics and Implementation:

The Secure5G framework involves the use of deep learning techniques to detect and mitigate security threats in network slices. The framework includes a security module that analyzes network traffic and identifies potential threats, and a response module that takes appropriate actions to mitigate the threats. The implementation of the framework involves the deployment of a centralized controller that manages the security module and the response module, and the use of a feedback loop to optimize the performance and security of the network slices.

3.6.3 Features:

The features of the Secure5G framework include the ability to detect and mitigate security threats in real-time, the ability to adapt to changing network conditions and requirements, and the ability to provide different levels of security and privacy to different applications and services. The framework also enables network operators to offer flexible pricing models, as resources can be allocated and de- allocated based on security requirements.

3.6.4 Evaluation:

The evaluation of the Secure5G framework involves several metrics, including security effectiveness, resource utilization, service quality, and scalability. Security effectiveness is measured by the ability of the framework to detect and mitigate security threats in network slices. Resource utilization is measured by the ability of the framework to optimize resource allocation while maintaining a high level of security and service quality. Service quality is measured by the ability of the framework to provide different levels of service quality and privacy to different applications and services. Scalability is measured by the ability of the framework to support a large number of network slices, each with different security requirements and characteristics.

3.7 Deep Learning and Blockchain Integration for Enhanced Security in 5G-Enabled IoT Networks [8]

(RESEARCH PAPER - 7)

3.7.1 Introduction:

This paper proposes a security framework for intelligent 5G-enabled IoT that combines deep learning and blockchain technologies. The framework aims to address the security challenges faced by IoT devices in 5G networks, including attacks on device identity, data privacy, and network availability. The framework uses deep learning to detect and mitigate security threats, and blockchain to provide a secure and decentralized platform for data storage and management.

3.7.2 Characteristics and Implementation:

The proposed framework involves the use of deep learning algorithms to detect and classify security threats in IoT devices and networks, and to generate alerts and responses in real-time. The framework also uses blockchain to create a secure and transparent platform for data management, authentication, and access control. The implementation of the framework involves the deployment of edge nodes and gateways that can communicate with IoT devices, and a centralized or decentralized blockchain network that can provide secure and reliable data storage and management.

3.7.3 Features:

The features of the proposed framework include the ability to detect and mitigate security threats in real-time, the ability to provide secure and decentralized data storage and management, the ability to support different levels of access control and privacy, and the ability to integrate with existing 5G network infrastructures. The framework also enables network operators to offer flexible pricing models, as resources can be allocated and de-allocated based on security and performance requirements.

3.7.4 Evaluation:

The evaluation of the proposed framework involves several metrics, including security effectiveness, resource utilization, service quality, and scalability. Security effectiveness is measured by the ability of the framework to detect and mitigate security threats in IoT devices and networks. Resource utilization is measured by the ability of the framework to optimize resource allocation while maintaining a high level of security and service quality. The framework's capacity to offer various levels of service quality and privacy to various IoT devices and apps serves as a gauge of service quality. The framework's capacity to accommodate a vast number of IoT devices and apps, each with a variety of security requirements and features, is a key indicator of its scalability.

3.8 Machine Learning-Driven Automation for 5G Network Slice

Control Functions [10] (RESEARCH PAPER - 8)

3.8.1 Introduction:

This paper proposes an automated approach to control the functions of 5G network slices using machine learning techniques. The goal is to enable faster and more efficient resource allocation and optimization for network slices. The proposed approach uses reinforcement learning to optimize the network slice functions, including the allocation of radio resources, computation resources, and storage resources.

3.8.2 Characteristics and Implementation:

The proposed approach involves the use of reinforcement learning algorithms to learn the optimal control policies for network slice functions. The algorithms use real-time data from the network to optimize the allocation of resources based on the performance requirements of the network slices. The implementation of the approach involves the deployment of a learning agent that interacts with the network and learns the optimal control policies over time. The agent can be deployed centrally or in a distributed way across the network, depending on the network topology and requirements.

3.8.3 Features:

The features of the proposed approach include the ability to optimize the allocation of resources in real-time, the ability to learn and adapt to changing network conditions and requirements, and the ability to provide efficient resource allocation and optimization for different types of network slices. The approach also enables network operators to offer flexible pricing models based on the performance and resource requirements of the network slices.

3.8.4 Evaluation:

The evaluation of the proposed approach involves several metrics, including the performance

of the network slices, the efficiency of resource allocation, and the scalability of the approach. The approach's capacity to satisfy the performance criteria of the network slices serves as a proxy for slices. The capacity of the strategy to distribute resources optimally while minimizing resource waste serves as a gauge of resource allocation efficiency. The approach's scalability is determined by its capacity to manage a large number of network slices and adjust to shifting network circumstances and requirements.

3.9 Enhancing 5G Radio Access Network Slicing Through Intelligent Resource Scheduling [3] (RESEARCH PAPER - 9)

3.9.1 Introduction:

For 5G radio access network (RAN) slicing, this study suggests an intelligent resource scheduling method that attempts to maximize the use of radio resources and enhance overall system performance. The suggested method makes use of machine learning techniques to forecast network slice traffic demands and dynamically assign radio resources to satisfy various network slice performance criteria.

3.9.2 Characteristics and Implementation:

The proposed approach involves two main stages: traffic demand prediction and resource allocation. In the traffic demand prediction stage, a machine learning model is trained to predict the traffic demand of network slices based on historical data and real-time network measurements. The anticipated traffic demand is utilized to dynamically distribute radio resources to various network slices in order to satisfy their performance needs. The method's implementation entails the installation of a centralized controller that communicates with the RAN nodes and distributes resources in accordance with the forecasts and performance specifications of network slices.

3.9.3 Features:

The features of the proposed approach include the ability to predict traffic demand and allocate resources dynamically, the ability to adapt to changing network conditions and traffic patterns,

and the ability to optimize the utilization of radio resources. The approach also enables network operators to offer flexible service-level agreements (SLAs) to different network slices based on their performance requirements.

3.9.4 Evaluation:

The evaluation of the proposed approach involves several metrics, including the resource utilization, network throughput, and the performance of network slices. The results show that the proposed approach outperforms traditional resource allocation schemes in terms of resource utilization and network throughput while meeting the performance requirements of different network slices. The approach also demonstrates the ability to adapt to changing traffic patterns and optimize the utilization of radio resources.

3.10 Exploring the Landscape of 5G Network Slicing: Challenges, Opportunities, and Use Cases [11] (RESEARCH PAPER - 10)

3.10.1 Introduction:

The challenges and opportunities that network slicing in 5G networks presents are discussed in this study. Enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (URLLC), and massive machine-type communications (mMTC) are just a few of the use cases that are covered in this article. It also covers the necessity for network slicing, its advantages, and the several use cases it can enable.

3.10.2 Characteristics and Implementation:

The ability to divide the network into various virtual networks, each with its own resources and administration tools, is one of the features of network slicing that are covered in the article. It also discusses the different ways that network slicing can be implemented, such as through the use of technologies like software-defined networking (SDN) and network function

virtualization (NFV).

3.10.3 Features:

The features of network slicing include the ability to provide customized services to different users or applications, the ability to optimize resource utilization, and the ability to enable flexible service-level agreements (SLAs) between network operators and their customers. Network slicing also allows for dynamic resource allocation, improved network efficiency, and reduced operational costs.

3.10.4 Evaluation:

The paper discusses the challenges and opportunities associated with network slicing, including the need for standardized interfaces, orchestration, and management frameworks. It also highlights the importance of security, privacy, and trust in network slicing, as well as the need for efficient resource allocation algorithms and effective service-level agreements. The paper concludes by discussing the potential impact of network slicing on various industries, such as healthcare, transportation, and manufacturing.

CHAPTER 4

PROJECT REQUIREMENTS SPECIFICATION

4.1 System Features

The Machine Intelligence Based Quality of Service (QoS) Framework is designed to optimize resource allocation and enable the creation and management of network slices with varying QoS requirements. The major features of the product include:

- **Resource allocation optimization:** The system leverages machine intelligence algorithms to optimize resource allocation in real-time. This ensures that network resources are allocated efficiently and effectively, thereby reducing network congestion thus, improving network performance.
- **QoS management:** The system enables the creation and management of network slices with varying QoS requirements. This allows network administrators to configure network slices according to specific traffic management policies and prioritize traffic based on the importance of the data.
- **Real-time monitoring:** The system provides real-time monitoring and management of network slices, enabling network administrators to detect and address issues as they arise. This ensures that network performance remains consistent and any issues can be resolved quickly.
- **User management:** The system enables network administrators to manage users and assign them to specific network slices based on their requirements. This ensures that users receive the appropriate level of service and that network resources are allocated optimally.
- **Scalability:** The system is designed to be scalable and can be deployed on a cloud-based server. This allows network administrators to manage their networks on a

larger scale and respond quickly to changing network demands.

- **Flexibility:** The system is designed to be flexible and customizable, enabling network administrators to configure network slices according to their specific requirements. This allows for greater control over network resources.

4.2 Operating Environment

The Machine Intelligence Based Quality of Service (QoS) Framework can be deployed on a variety of hardware platforms and operating systems. The system's software components can be installed and configured on a cloud-based server, which can be accessed remotely by network administrators and service providers. The following are the operating environment requirements for the system:

- **Hardware platform:** The system can be deployed on a variety of hardware platforms, including servers, workstations, and laptops. The hardware platform should meet the minimum requirements for running the system's software components, including CPU, RAM, and storage space.
- **Operating system:** The system's software components are compatible with multiple operating systems, including Windows, Linux, and macOS. The operating system version should be up-to-date and compatible with the software components.
- **Software components:** The system's software components include machine intelligence algorithms, network slicing modules, real-time monitoring and management tools, and user management tools. These components should be installed and configured according to the system's documentation and requirements.
- **Network infrastructure:** The system requires a stable and reliable network infrastructure to function properly. The network infrastructure should be capable of handling the data traffic generated by the system and should have sufficient bandwidth and capacity to support the system's requirements.

It is designed to operate in a flexible and scalable environment. The system's software components can be installed and configured on a variety of hardware platforms and operating systems, enabling users to deploy the system according to their specific requirements. The system's network infrastructure requirements are minimal, and the system can be accessed remotely, making it a versatile and efficient solution for network administrators and service providers.

4.3 General Constraints, Assumptions and Dependencies

There are several constraints, assumptions, and dependencies that developers of the Machine Intelligence Based Quality of Service (QoS) Framework should be aware of. These include:

- **Regulatory policies:** The system should comply with all relevant regulatory policies, such as data protection and privacy laws. This may limit the choices available to developers in terms of the system's design and implementation.
- **Hardware limitations:** The system's performance may be affected by hardware limitations, such as signal timing requirements or insufficient processing power. Developers may need to optimize the system's algorithms and architecture to accommodate these limitations.
- **Limitations of simulation programs:** The accuracy and reliability of the system's simulation programs may be limited by factors such as the quality and availability of data. Developers may need to incorporate additional data sources or improve data processing methods to improve the accuracy of the simulations.
- **Interfaces to other applications:** The system may need to interface with other applications, such as network management tools or billing systems. Developers may need to ensure that the system's interfaces are compatible with these applications and that data can be transferred seamlessly between systems.

- **Parallel operations:** The system may need to perform parallel operations, such as managing multiple network slices simultaneously. Developers may need to optimize the system's architecture and algorithms to ensure efficient parallel processing.
- **Criticality of application:** The system's performance may be critical for certain applications, such as emergency services or healthcare. Developers may need to prioritize the system's performance and reliability for these critical applications.
- **Safety and security considerations:** The system may handle sensitive data or operate in environments with strict safety or security requirements. Developers may need to implement additional security measures, such as encryption or access controls, to protect sensitive data and ensure system safety.

4.4 Risks

During the development of the Machine Intelligence Based Quality of Service (QoS) Framework, there are several risks that should be considered. These risks include:

- **Resource requirements:** The system may require significant computing resources, such as processing power and storage space, to function effectively. Failure to allocate sufficient resources may result in system performance issues and reduce the effectiveness of the QoS framework.
- **Technical complexity:** The QoS framework is a technically complex system that requires expertise in machine learning, networking, and real-time monitoring. Failure to manage technical complexity may result in delays, cost overruns, and reduced functionality.
- **Integration with existing systems:** The QoS framework may need to integrate with existing network infrastructure, including legacy systems and third-party tools. Integration may be challenging, and failure to integrate effectively may result in

compatibility issues, reduced functionality, and increased costs.

- o **Security risks:** The QoS framework may be vulnerable to security risks, such as hacking, data breaches, and denial-of-service attacks. Developers must ensure that the system is secure and follows best practices for data protection and security.
- o **User adoption:** The QoS framework may be complex and require training for users. Failure to ensure user adoption may result in low usage rates, reduced functionality, and reduced return on investment.
- o **Regulatory compliance:** The QoS framework may be subject to regulatory compliance requirements, such as data protection and privacy laws. Failure to comply with regulations may result in legal consequences and damage to the reputation of the organization.

These risks must be identified, evaluated, and addressed throughout the development process of the Machine Intelligence Based QoS Framework. Failure to manage these risks effectively may result in delays, cost overruns, and reduced functionality.

4.5 Functional Requirements

- **Slice creation:** The system should allow for the creation of network slices with specific requirements such as latency, throughput, and bandwidth.
- **Resource allocation:** The system should allocate the necessary resources such as compute, storage, and network resources to each slice based on their individual requirements.
- **Slice isolation:** The system should ensure that each slice is isolated from other slices to prevent interference and maintain security.
- **Dynamic scaling:** The system should be able to dynamically adjust the resources allocated to each slice based on changing requirements or demand.
- **Quality of service (QoS) management:** The system should prioritize traffic based on QoS requirements and ensure that each slice meets its performance

objectives.

- **Fault tolerance:** The system should be resilient to faults and failures, and should be able to detect and recover from any issues that may arise.
- **Monitoring and analytics:** The system should provide real-time monitoring and analytics to enable operators to identify issues, optimize performance, and troubleshoot problems.
- **Error handling and recovery:** The error handling and recovery mechanisms are incorporated so that the system can continue to operate even in the event of a failure or error which includes automated fail over, backup and restore procedures, and proactive monitoring and alerting.

4.6 Hardware Requirements

The system should be able to run on a variety of hardware platforms, including desktops, laptops, servers, and mobile devices. The hardware components of the system should be connected through a standard communication protocol, such as Ethernet or Wi-Fi, and should be compatible with the operating system that the software product is running on.

The hardware components of the system should also meet certain minimum specifications in terms of processing power, memory, and storage capacity. The following are the minimum hardware requirements for the system:

- **Processor:** The system should be able to run on a processor with a clock speed of at least 2 GHz.
- **Memory:** The system should have a minimum of 8 GB of RAM to ensure that it can process large amounts of data quickly.
- **Storage:** The system should have a minimum of 100 GB of storage space to store the software product, as well as any data or files that are generated by the system.
- **Network interface:** The system should be connected to a network through a

standard communication protocol, such as Ethernet or Wi-Fi, to allow for communication with other devices on the network.

- **Operating system:** The system should be compatible with the operating system that the software product is running on, such as Windows or Linux.

Overall, the hardware requirements for the system should be designed to ensure that the software product can run efficiently and effectively on a variety of hardware platforms and can process large amounts of data quickly.

4.7 Software Requirements

- **Name and Description:** The software product that will be used in this project is a machine learning framework, which will be used to train and deploy machine learning models.
- **Version/Release Number:** The latest stable version of the software product will be used in the project.
- **Operating Systems:** The software product should be compatible with a variety of operating systems, including Windows, Linux, and MacOS.
- **Tools and Libraries:** Python, Tensorflow, Scikit-learn, Pandas, Numpy, Matplotlib.

4.8 Performance Requirement

The performance requirements for the Machine Intelligence Based Quality of Service (QoS) Framework are as follows:

- o **Reliability:** The system must be reliable and provide consistent results under normal operating conditions. The system should be able to handle a large

number of requests without crashing or failing.

- o **Availability:** The system will be available for use at all times, except during scheduled maintenance periods. The system should be able to handle high traffic volumes without slowing down or crashing.
- o **Response Time:** The system will respond quickly to user requests and provide results within a reasonable time frame. The response time should be consistent, regardless of the number of users or the complexity of the request.
- o **Scalability:** The system will be scalable to handle increased traffic volumes and additional users. It should be able to handle increasing workloads without degradation in performance.

CHAPTER 5

SYSTEM DESIGN

5.1 Current System

As per the proposed approach for resource allocation in network slicing, there is no specific current system being enhanced, replaced, or upgraded. However, if we consider the current approaches for resource allocation in network slicing, they mostly rely on heuristic-based methods or rule-based systems, which have limited capabilities to adapt to changing network conditions and lack the ability to learn from past experiences. These approaches also require a significant amount of manual intervention and do not take advantage of the vast amount of data available in the network. Hence, the proposed approach aims to address these limitations by leveraging machine learning techniques such as supervised and reinforcement learning.

5.2 Design Goals

1. **Improved Resource Allocation:** The primary goal of the proposed approach is to achieve improved resource allocation in network slicing, leading to better performance and efficient utilization of resources.
2. **Scalability and Adaptability:** The proposed approach should be scalable and adaptable to different network topologies and varying workload conditions. It should be able to adjust to changes in the network environment and the requirements of different applications and devices.
3. **Performance:** The proposed approach should be designed to deliver high performance in terms of throughput, latency, and packet loss. It should be able to handle high traffic loads and ensure fast and reliable service delivery.
4. **Security and Privacy:** The proposed approach should ensure the security and privacy of user data and prevent unauthorized access and malicious attacks. It should also comply with privacy regulations and protect user privacy.

-
5. **User Experience:** The proposed approach should provide a good user experience, with intuitive interfaces and easy-to-use features. It should also ensure a consistent look and feel across different devices and platforms
 6. **Cost-effectiveness:** The proposed approach should be cost-effective, with minimal hardware and software requirements and low operational costs. It should also provide a good return on investment and deliver business value.

5.3 Architecture Choices

- In designing the proposed solution for the resource allocation in network slicing, various architecture choices were considered. These choices included supervised learning, deep reinforcement learning, and hybrid approaches combining both techniques.
- The supervised learning approach was initially considered as it is a well-established method in machine learning, and there is a large body of literature on its application in resource allocation problems. However, one of the drawbacks of the supervised learning approach is that it requires a large amount of labeled data, which can be costly and time-consuming to obtain.
- On the other hand, deep reinforcement learning (DRL) has shown promising results in solving complex decision-making problems, such as resource allocation in network slicing. DRL can learn to optimize the allocation of resources based on the feedback from the environment. One of the benefits of DRL is that it can learn from both labeled and unlabeled data, which can help address the issue of limited labeled data.
- However, the drawback of DRL is that it can be computationally expensive and requires a significant amount of training time. Moreover, the performance of DRL can be highly dependent on the choice of hyperparameters, and tuning them can be a challenging task.
- To overcome the limitations of both supervised learning and DRL, a hybrid approach was considered. In this approach, we can use supervised learning to pre-train a deep neural network on a large dataset and then fine-tune the network using DRL on a smaller labeled dataset. This can help leverage the benefits of both approaches and improve the performance of the model.
- In summary, while the supervised learning approach is well-established and easy to

implement, it has limitations in terms of the requirement of large labeled data. DRL can overcome this limitation but can be computationally expensive and highly dependent on the choice of hyperparameters. The hybrid approach can combine the benefits of both approaches and can potentially lead to better performance.

5.4 High Level System Design

Design Description

We are first explaining how our machine learning model plays its role in the process of a user being allocated the right network slice for their requirement. So, the major modules at a high level involved in this process are:

- **The Network Administrator:** This entity is responsible for allocation of resources on the network slice and creates the right slice for the user.
- **The Service Provider:** This module is the connecting link between the user application and network admin. The user can only describe their application and what are the necessary components so the service provider helps deliver the correct requirements to the machine intelligence aid with the help of which the network administrator understands the right slice to be assigned for the user.
- **Machine Intelligence:** The most efficient algorithm is already fed into this entity which receives the application of the user as the input and verifies if the slice being assigned is optimal and sufficient for the user or not.

5.5 5G Network Slicing Architecture Overview

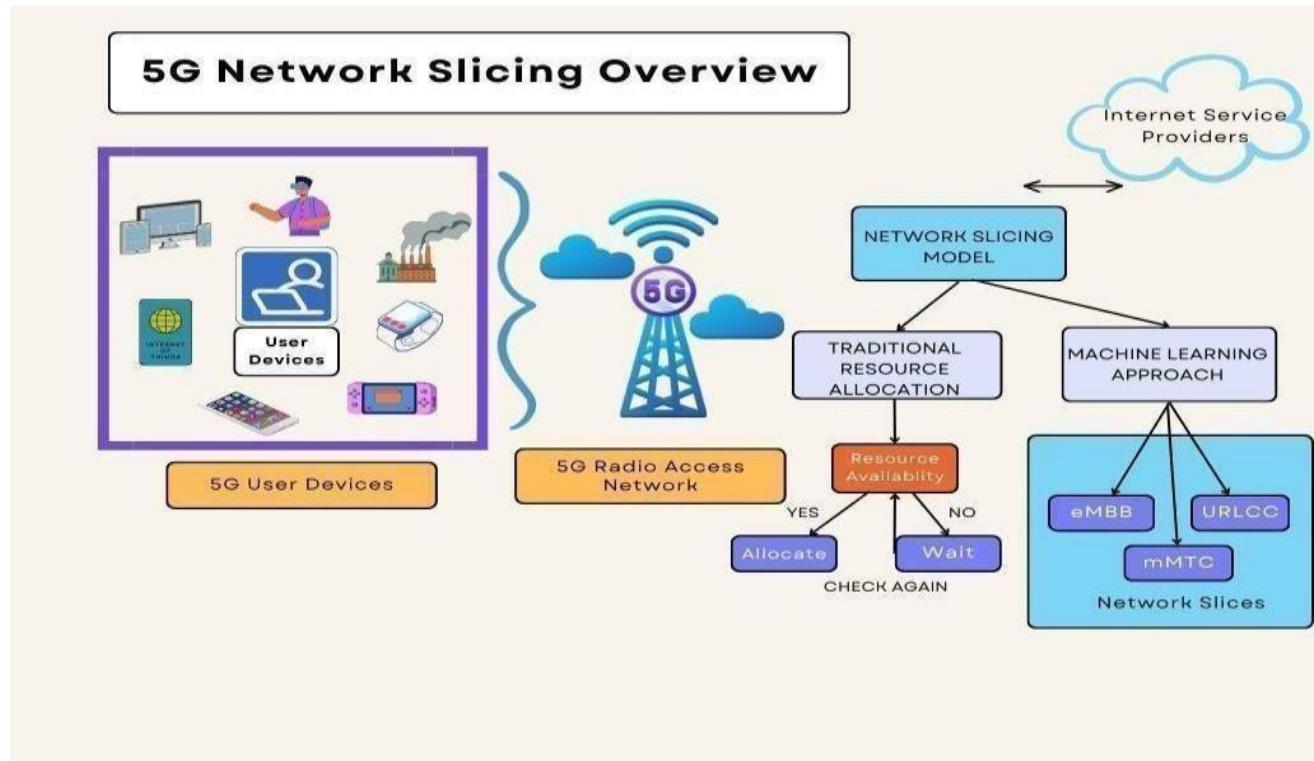


Fig – 5.5.1: 5G Network Slicing Overview diagram

NAME OF SLICE TYPE	SLICE REQUIREMENTS	LATENCY	THROUGHPUT	THEIR APPLICATIONS
eMBB (enhanced Mobile Broadband)	High data rates and low latency	1 ms - 10 ms	1 Gbps - 10 Gbps	Video streaming, virtual reality
mMTC (Massive Machine Type Communication)	High connectivity and low power consumption	10 ms - 100 ms	100 kbps - 1 Mbps	Internet of Things (IoT) devices, smart city solutions
uRLLC (Ultra-Reliable Low-Latency Communication)	Extremely low latency and high reliability	1 millisecond !	100 kbps - 1 Mbps	Autonomous vehicles, remote surgery

Fig – 5.5.2: Details about the types of network slices

5.6 Master Class Diagram

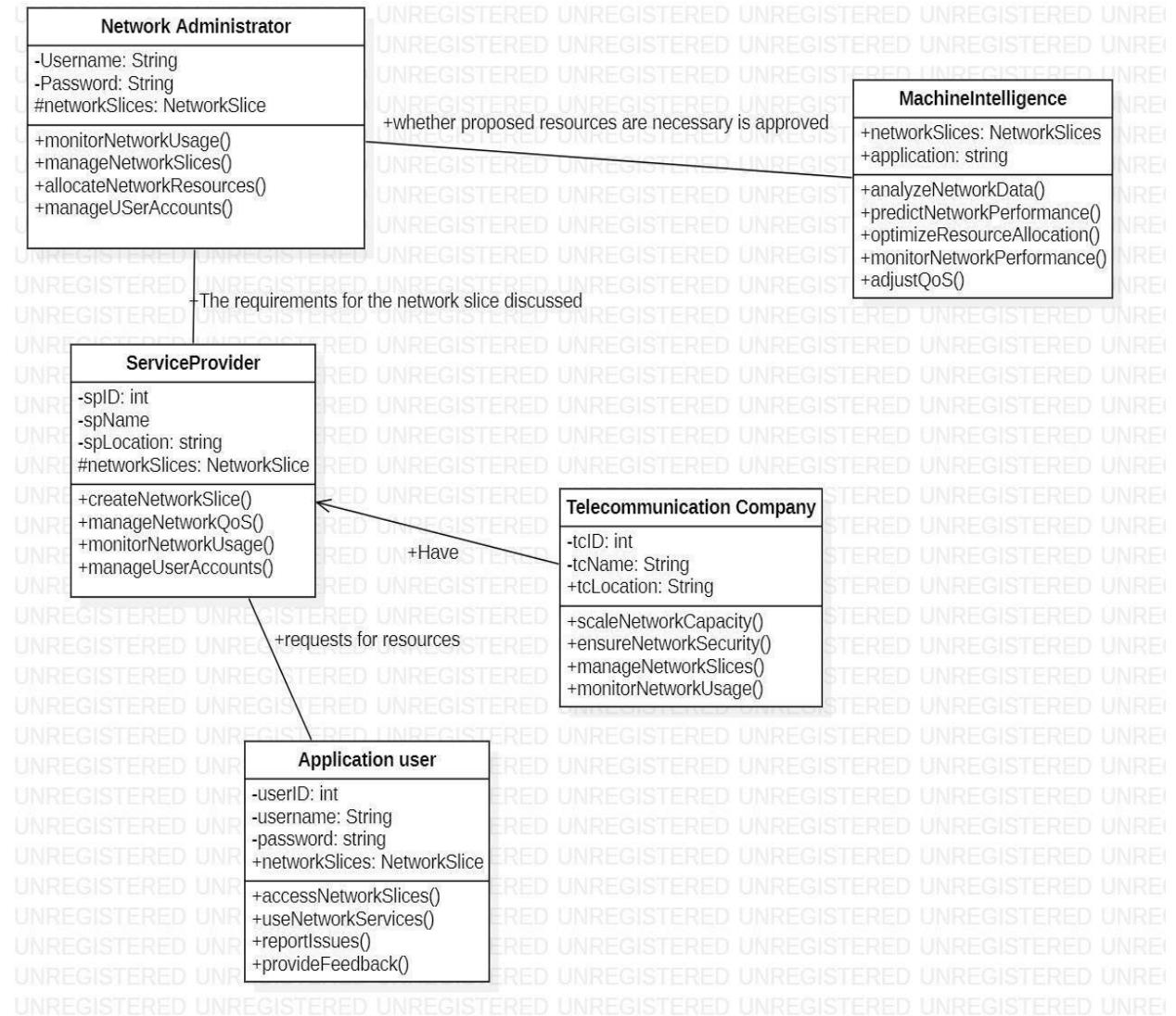


Fig 5.6.1 : MASTER CLASS Diagram

In fig 5.5.1 **NetworkAdmin** and **ServiceProvider** classes are responsible for managing the network and its resources. The **ServiceProvider** class has methods for creating network slices, managing network QoS, monitoring network usage, and managing user accounts. The **NetworkAdminclass** has methods for monitoring network usage, managing network slices, allocating network resources, and managing user accounts.

The **MachineIntelligence** class is responsible for analyzing network data, predicting

network performance, optimizing resource allocation, and monitoring network performance. The **TelecommunicationCompany** class has methods for scaling network capacity, ensuring network security, managing network slices, and monitoring network usage.

The **Application user** class represents users of the network, who have access to network slices and use network services. The EndUser class has methods for accessing network slices, using network services, reporting issues, and providing feedback.

All classes have a member variable called **networkSlices**, which is an array of Network Slice objects. The NetworkSlice class is not shown in the diagram, but it is assumed to exist and contain information about specific network slices.

5.7 Packaging and Deployment Diagram

The diagram below gives a high level idea on the position of the machine intelligence tool and its vitality in the main frame application of ensuring quality of service to the users when network slicing takes place.

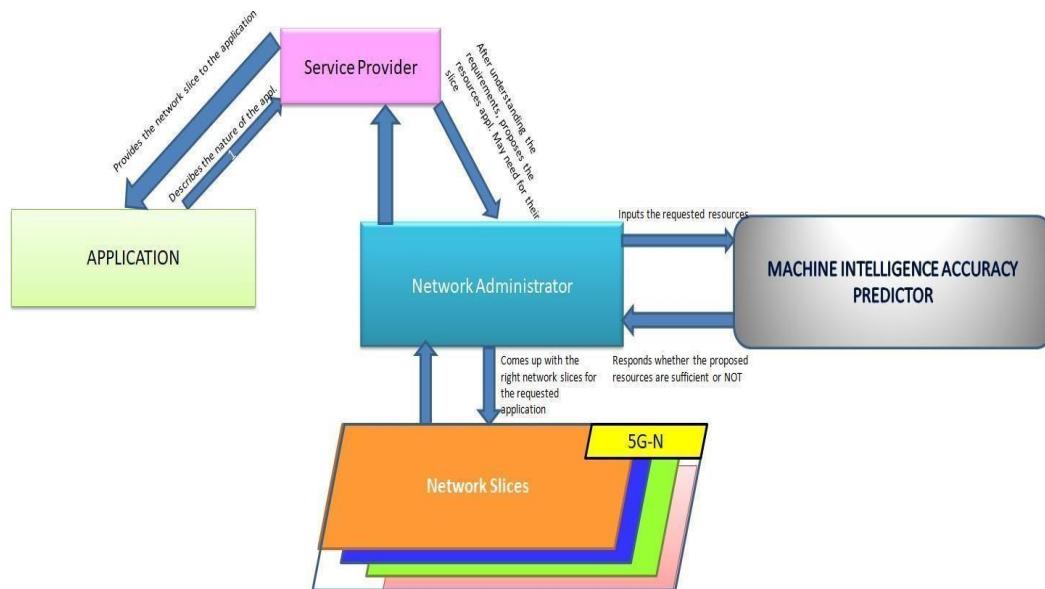


Fig 5.7.1: Deployment Diagram

5.8 Help

This project is focused on developing the Machine Intelligence Accuracy predictor which, based on

the dataset it has been trained on, indicates whether the resources being demanded by the user application is sufficient for running the application effectively or not. The Machine Intelligence model will require further assistance from the Network Admin if it needs to be extended in determining the right quantity of resources.

5.9 Design Details

- **Novelty** : It aims to improve the QoS for dynamic resource allocation using a combination and reinforcement learning. This approach is relatively novel, and there may be opportunities to further innovate and improve the approach.
- **Innovativeness** : The approach involves the use of data augmentation, transfer learning, and ensemble learning techniques to improve the performance of the model. These techniques are innovative and can help to improve the accuracy and robustness of the model.
- **Interoperability** : It will need to be compatible with existing network infrastructures and communication protocols. This will require careful consideration of interoperability issues and may require modifications to existing systems.
- **Performance** : It will depend on the quality and quantity of the data, as well as the accuracy of the models. The use of data augmentation, transfer learning, and ensemble learning techniques can help to improve the performance of the model.
- **Security** : It will need to be secure to prevent unauthorized access to sensitive data and protect against malicious attacks. This will require the use of appropriate encryption and security protocols.
- **Reliability** : It will need to be reliable to ensure that it can provide consistent and accurate results over time. This will require the use of robust models and testing procedures to ensure that the system is reliable.

-
- **Maintainability** : It will need to be maintainable to ensure that it can be updated and modified over time. This will require careful documentation and version control procedures.
 - **Portability** : It will need to be portable to allow it to be used in different environments and systems. This will require careful consideration of platform dependencies and system requirements.
 - **Legacy to modernization**: It may require the modernization of existing systems to ensure that they are compatible with the new approach. This may require the use of legacy integration techniques and careful testing procedures.
 - **Reusability** : It should be designed to be reusable to allow it to be applied to different use cases and scenarios. This will require the use of modular design principles and careful documentation.
 - **Application compatibility**: It should be compatible with existing applications and systems to ensure that it can be integrated into existing workflows. This will require careful consideration of application dependencies and compatibility issues.

5.10 RESOURCE UTILIZATION

1. **Resource Management Systems**: The proposed approach involves resource allocation in 5G networks, which requires effective management of network resources such as bandwidth, latency, and processing power. Resource management systems are critical to ensure that resources are allocated efficiently and effectively.
2. **Cloud Computing Platforms**: The use of cloud computing platforms can

provide significant benefits for resource allocation in 5G networks. Cloud platforms can provide on-demand access to computing resources, which can help meet the dynamic resource requirements of different applications.

3. **Network Function Virtualization (NFV):** NFV is an approach to network architecture that involves virtualizing network functions, such as routing, firewalling, and load balancing. NFV can help improve resource utilization in 5G networks by allowing network functions to be deployed more efficiently.
4. **Edge Computing:** Edge computing involves processing data closer to the source of the data, rather than sending it to a centralized data center. This can help reduce latency and improve performance, which is particularly important for applications that require real-time data processing, such as healthcare.
5. **Machine Learning Platforms:** The proposed approach involves the use of machine learning algorithms for resource allocation. Machine learning platforms provide a range of tools and services for developing, deploying, and managing machine learning models.
6. **DevOps Processes:** The development and deployment of machine learning models require a range of processes, including version control, continuous integration, and continuous deployment. DevOps processes can help streamline the development and deployment of machine learning models.

CHAPTER 6

PROPOSED METHODOLOGY

6.1 Approach 1: Using Traditional Resource Allocation Algorithms

The traditional 5G network slicing approach employs diverse resource allocation methods, with our discussion centered on two specific algorithms.

- A. **Dynamic Allocation:** This real-time resource allocation method dynamically adjusts resources based on evolving application demands, ensuring adaptive and efficient utilization.
- B. **Priority-based allocation:** This resource allocation method assigns resources based on application priority levels, prioritizing higher-priority applications for preferential access. The strategy aligns resource allocation with the criticality or importance of each application in the system.

By combining both the above algorithms, we integrated a hybrid approach that incorporates both dynamic resource allocation, responsive to the real-time demands of applications, and priority-based allocation, ensuring resources are assigned in accordance with application priority levels. This synergy creates a comprehensive resource allocation strategy that adapts to evolving requirements while strategically prioritizing higher-criticality applications for optimal system performance. In a 5G network simulation, the Base Station is central, overseeing coverage areas, capacities, and network slices, dynamically allocating resources for client connections. Clients, as user devices, exhibit modeled behaviors and connect/disconnect from slices, consuming resources. The Coverage entity defines a base station's spatial extent. The Distributor generates random values for mobility patterns and usage frequencies. Network Slices, within the Slice entity, manage users, bandwidth, and capacities. Resource containers, represented by the Container

entity, model capacity within slices for resource allocation. The KDTree, implementing a k-d tree, is a crucial tool for spatial searches. It enhances efficiency by swiftly identifying the closest base stations to clients, facilitating optimized connectivity within the simulated network environment. Together, these interconnected components form a comprehensive and dynamic simulation framework, enabling the exploration of various scenarios and behaviors within the context of a 5G network. These are the results obtained after performing a hybrid approach:

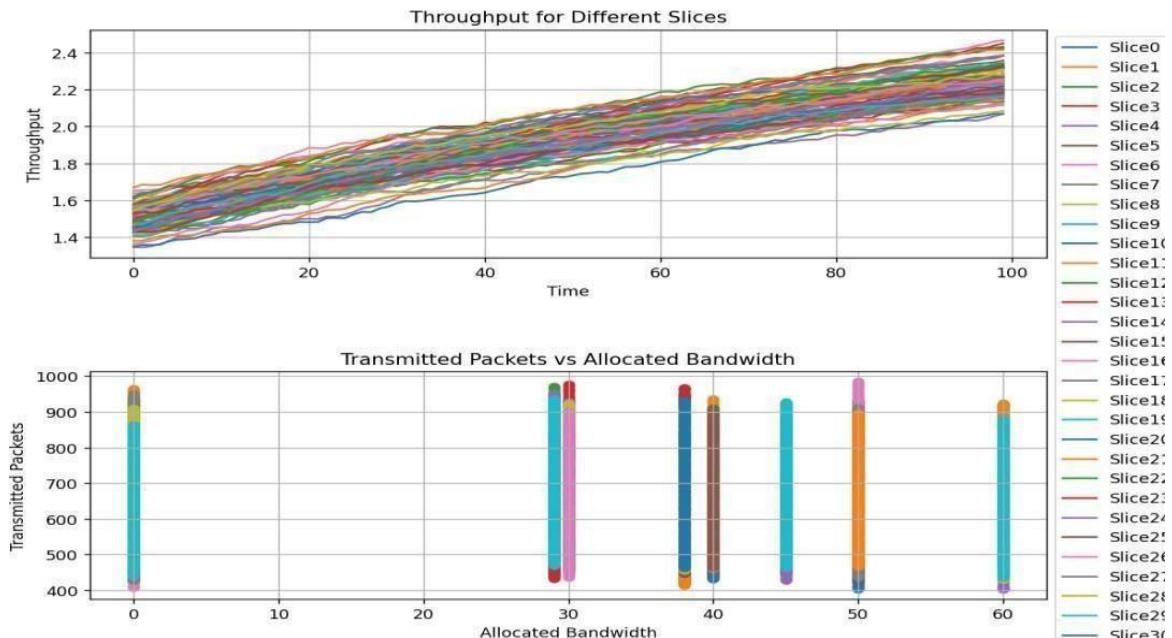


Fig – 6.1.1: Priority Based Allocation

Fig. 6.1.1 showcases prioritized resource distribution based on priority levels. Higher-priority slices receive increased resources, ensuring critical segments receive the necessary allocation.

Pseudocode:

```

class WFQResourceAllocator:
    def __init__(self, slices, switches):
        self.slices = slices
        self.switches = switches

    def allocate_resources(self):
        total_capacity = sum([switch.capacity for switch in self.switches])

        for network_slice in self.slices:
            required_bandwidth = network_slice.qos_requirements['bandwidth']
            allocated_bandwidth = int(total_capacity * (required_bandwidth / total_capacity))

            for switch in self.switches:
                if allocated_bandwidth <= switch.capacity:
                    switch.capacity -= allocated_bandwidth
                    network_slice.allocated_bandwidth = allocated_bandwidth
                    print(f"WFQResourceAllocator--Allocated {allocated_bandwidth} bandwidth to {network_slice.name}")
                    break
                else:
                    print(f"WFQResourceAllocator--Insufficient bandwidth to allocate to {network_slice.name}")

```

The **class WFQResourceAllocator** allocates bandwidth to network slices based on a Weighted Fair Queueing (WFQ) approach, considering the total capacity of switches. In the `allocate_resources` method, it calculates the total capacity by summing the capacities of all switches and then allocates bandwidth to network slices proportionally. It ensures that the allocated bandwidth doesn't exceed the available capacity of switches and logs an error message if insufficient bandwidth is available.

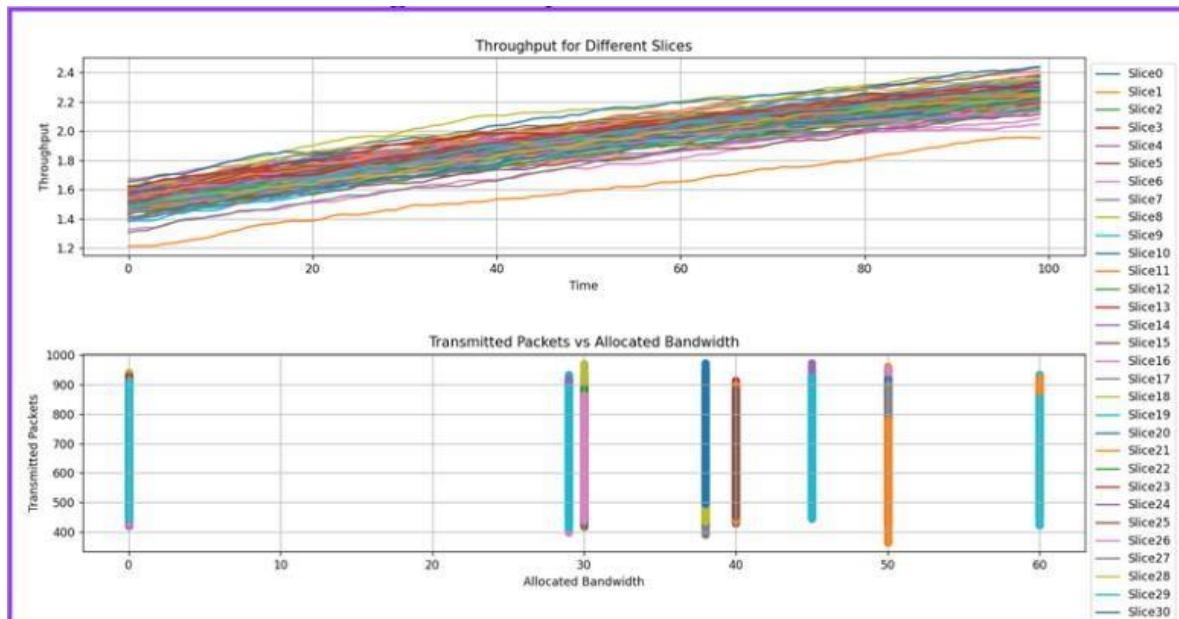


Fig – 6.1.2 Weighted Fair Queue Allocation

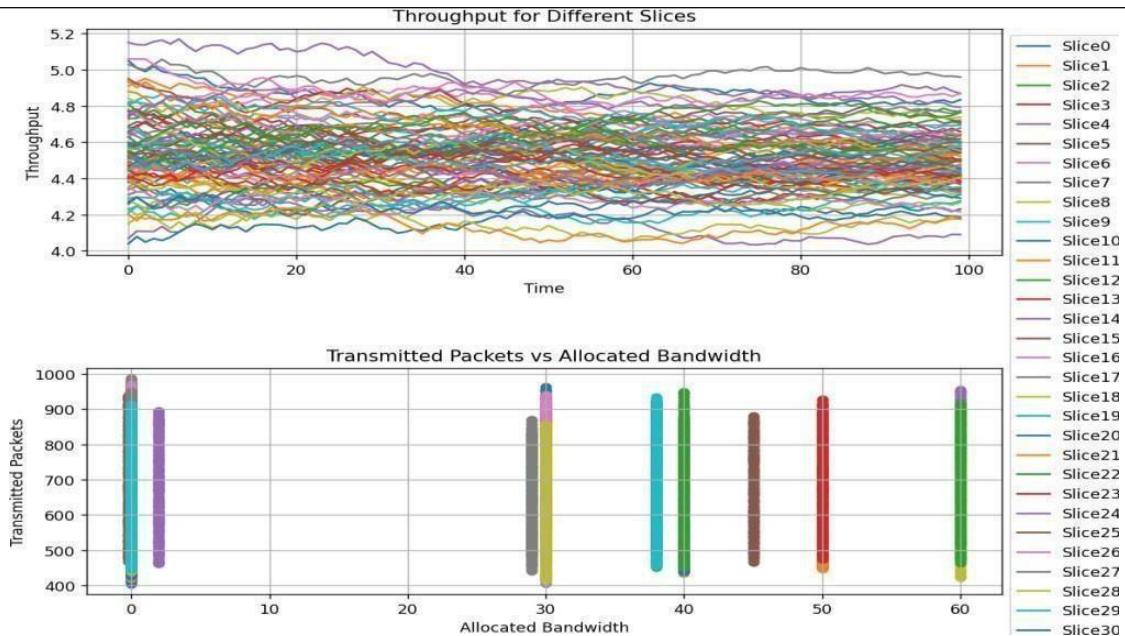

Fig – 6.1.3: Dynamic Resource Allocation

Fig.6.1.3. demonstrates the dynamic adaptation to evolving network conditions. It visualizes real-time adjustments in resource allocation to slices, catering to their individual requirements. This promotes adaptability and responsiveness in the allocation system

6.2 Approach 2: Using Machine Learning Algorithms

1) Random Forest Classifier:

- Ensemble learning method using a collection of decision trees.
- Robust to overfitting and handles missing values well.

2) AdaBoost Classifier:

- Boosting algorithm that combines weak learners into a strong one.
- Adaptive, focuses on misclassified samples.

3) Gradient Boosting Classifier:

- Sequentially builds trees, correcting errors of the previous ones.

- Robust to outliers and handles different types of data well.

4) Extra Trees Classifier:

- Similar to Random Forest but builds trees with random splits.
- Fast training and prediction, suitable for large datasets.

5) Bagging Classifier:

- Reduces variance and overfitting by training multiple models.
- Improves stability and generalization, works well with unstable models.

6) Decision Tree Classifier:

- Simple and interpretable model for classification and regression.
- Prone to overfitting but suitable when interpretability is crucial.

7) MLP Classifier (Multi-Layer Perceptron):

- Deep learning model capable of learning complex relationships.
- Suitable for complex classification and regression tasks.

8) Stacking Classifier:

- Ensemble learning technique that combines predictions from multiple models.
- Uses a meta-model to make the final prediction.

9) Voting Classifier:

- Combines predictions from multiple models by majority voting (hard voting) or weighted voting (soft voting).
- Ensemble method for improved overall performance.

10) LSTM

- Through sequential learning of network parameters, this model harnesses historical data patterns of Packet Loss Rate, Packet delay, slice Type, Time, and LTE/5g Category.
- Its sliding window approach and extensive epoch-based training enable precise predictions of future resource demands, empowering proactive and

adaptive resource allocation strategies in dynamic 5G network slicing scenarios.

6.3 Project Workflow

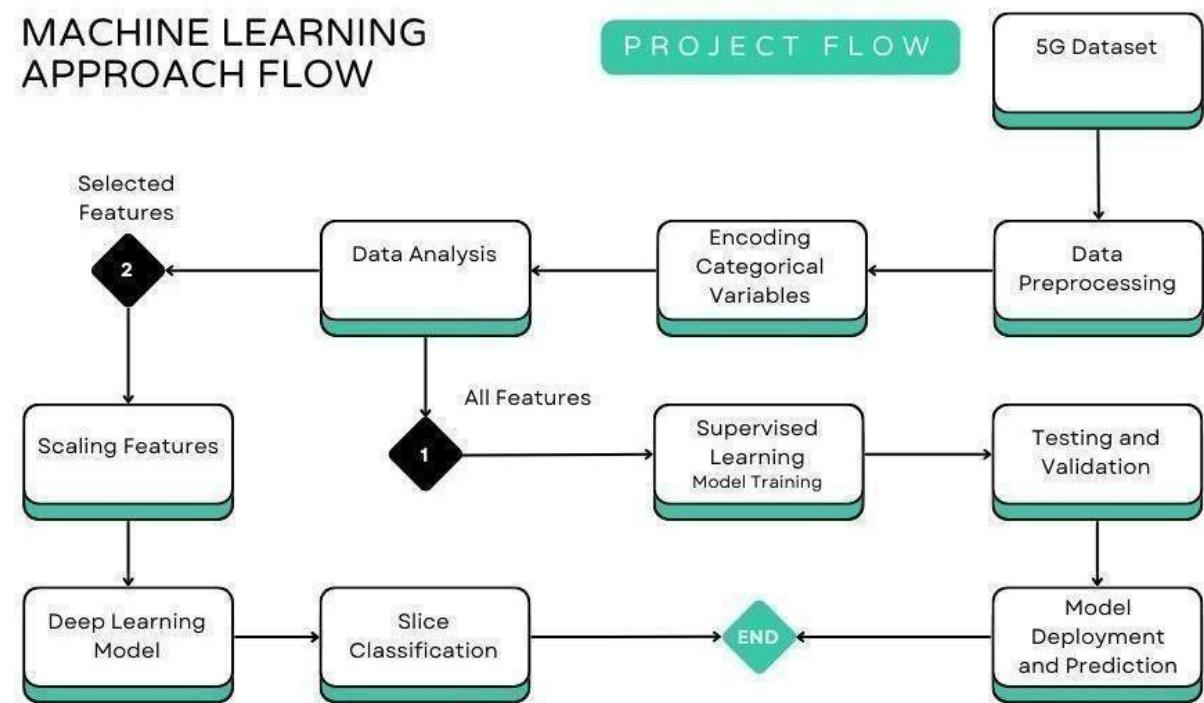


Fig – 6.3.1: Machine Learning Approach Workflow

CHAPTER 7

IMPLEMENTATION AND PSEUDOCODE

7.1 TRADITIONAL RESOURCE ALLOCATION ALGORITHMS

1. DYNAMIC CAPACITY ALLOCATOR

```

class DynamicCapacityAllocator:
    def __init__(self, slices, switches):
        self.slices = slices
        self.switches = switches

    def allocate_resources(self):
        for network_slice in self.slices:
            required_bandwidth = network_slice.qos_requirements['bandwidth']

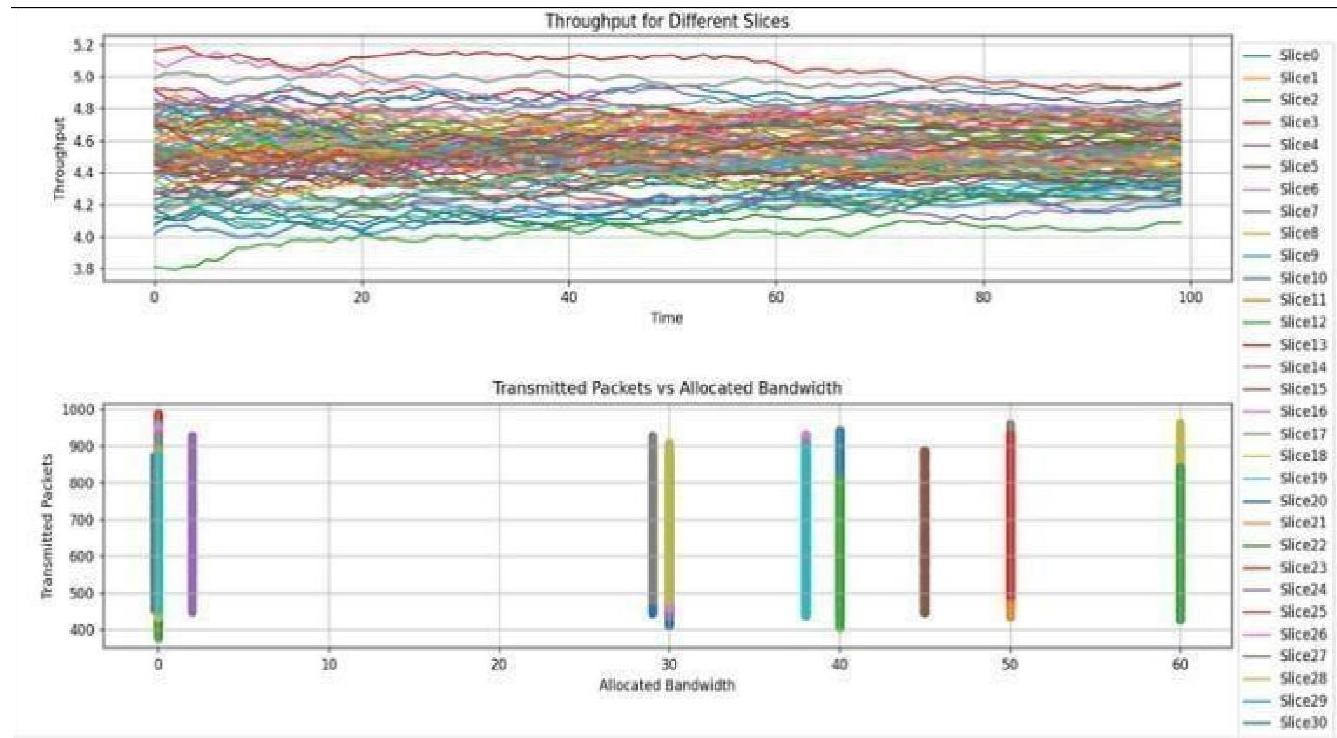
            for switch in self.switches:
                allocated_bandwidth = min(required_bandwidth, switch.capacity)
                switch.capacity -= allocated_bandwidth
                network_slice.allocated_bandwidth = allocated_bandwidth
                print(f"DynamicCapacityAllocator--Allocated {allocated_bandwidth} bandwidth to {network_slice.name}")
                if allocated_bandwidth >= switch.capacity * 0.8:
                    switch.capacity = switch.capacity - (allocated_bandwidth * 0.1)
                break
            else:
                print(f"DynamicCapacityAllocator--Insufficient bandwidth to allocate to {network_slice.name}")

    def deallocate_resources(self):
        for network_slice in self.slices:
            allocated_bandwidth = network_slice.allocated_bandwidth
            for switch in self.switches:
                if switch.capacity + allocated_bandwidth <= switch.initial_capacity:
                    switch.capacity += allocated_bandwidth
                    network_slice.allocated_bandwidth = 0
                    print(f"DynamicCapacityAllocator--Deallocated {allocated_bandwidth} bandwidth from {network_slice.name}")
                    break
                else:
                    print(f"DynamicCapacityAllocator--Unable to deallocate bandwidth from {network_slice.name}")

```

Fig – 7.1.1: Dynamic Capacity Allocation Pseudocode Diagram

Dynamic Capacity Allocator manages the allocation and deallocation of network bandwidth between network slices and switches. The `allocate_resources` method allocates bandwidth to slices while monitoring switch capacity, potentially reducing it by 10% if bandwidth usage exceeds 80%. The `deallocate_resources` method returns bandwidth to switches and logs an error if it can't deallocate without exceeding the initial capacity.


Fig – 7.1.2: Dynamic Allocation Results

2. PRIORITY BASED ALLOCATION

```
class PriorityBasedAllocator:
    def __init__(self, slices, switches):
        self.slices = slices
        self.switches = switches

    def allocate_resources(self):
        priority_slices = sorted(self.slices, key=lambda slice: slice.qos_requirements['priority'], reverse=True)

        for network_slice in priority_slices:
            required_bandwidth = network_slice.qos_requirements['bandwidth']

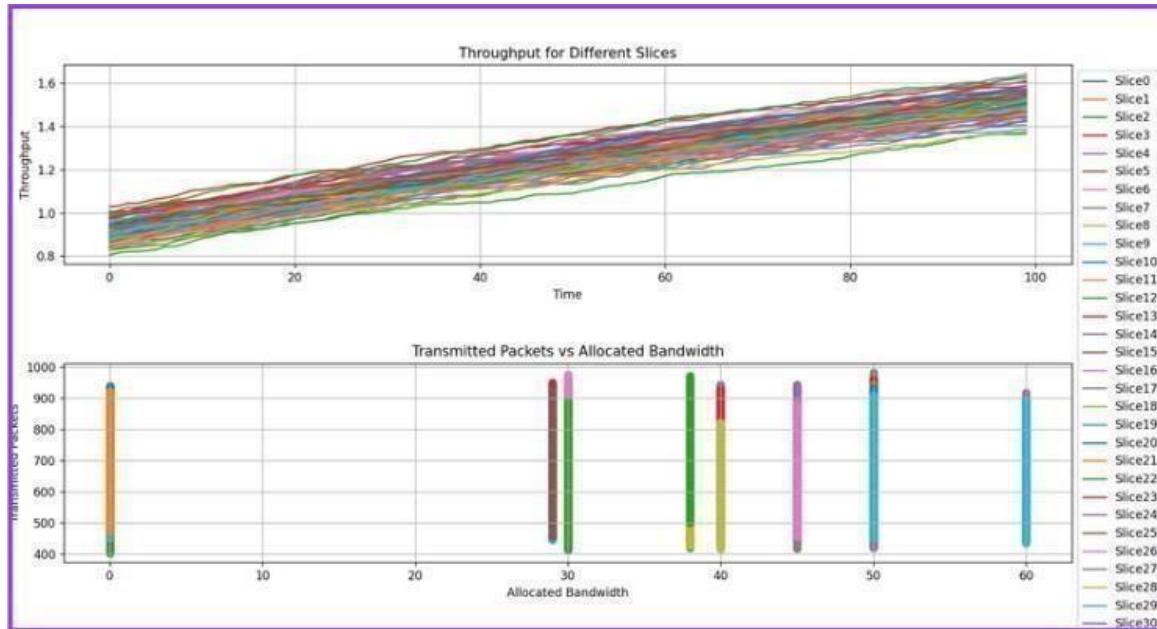
            for switch in self.switches:
                if required_bandwidth <= switch.capacity:
                    switch.capacity -= required_bandwidth
                    network_slice.allocated_bandwidth = required_bandwidth
                    print(f"PriorityBasedAllocator--Allocated {required_bandwidth} bandwidth to {network_slice.name}")
                    break
                else:
                    print(f"PriorityBasedAllocator--Insufficient bandwidth to allocate to {network_slice.name}")
```

Figure 7.1.5 : Priority-based Allocation pseudocode

PriorityBasedAllocator that allocates bandwidth to network slices based on their priority. It sorts the slices by priority in decreasing order.

In the allocate_resources method, it allocates bandwidth to network slices starting from the

highest priority slices. It checks if the required bandwidth is available in switches and logs an error message if there's insufficient bandwidth for a slice.



The three provided resource allocation algorithms cater to different scenarios and priorities:

- **Dynamic Capacity Allocator:** Best for dynamic environments with fluctuating bandwidth needs, such as real-time streaming services.
- **Priority Based Allocator:** Suitable for applications with strict priority requirements, like emergency services or mission-critical systems.

Algorithm	Slices allocated(%)	Fully allocated slices	Remaining slices	Remarks
Priority-based allocation	70 %	63	27	Efficiently prioritized slices based on specified priorities.
Weighted fair queuing	80%	72	18	Allocated resources based on weight factors, giving priority to slices with higher weights.
Proportional fairness	100%	90 (partially filled requirements)	0	Ensured fair resource distribution by allocating 60 - 70% of resource requirements for all slices.
Dynamic (over time) (Dynamic Resource allocation algorithm)	100%	90 (fully filled requirements)	0	Dynamically allocated and deallocated resources over time based on changing demand and network conditions.

TABLE 7.1.1: Traditional algorithms

7.2 Choice of Selection for Testing in 5G Architecture with Simulation

Dynamic Resource Allocation is the preferred choice for the 5G architecture due to its adaptability to changing conditions, efficient resource utilization, and responsiveness to demand fluctuations, and the ability to optimize Quality of Service (QoS). In a dynamic and diverse 5G environment, this approach ensures fair and effective distribution of resources across different slices, making it well-suited for the network's varying and evolving demands. While Weighted Fairness, Priority-Based Allocation, and Proportional Fairness have merits, Dynamic Resource Allocation's flexibility and responsiveness make it better aligned with the dynamic nature of 5G networks. The specific choice depends on the unique characteristics and

goals of the 5G network under consideration.

Let's setup the 5g architecture for Simulation with python:

1. Base station:

Purpose: Represents the base station in a 5G network.

Functionality:

Manages coverage area, capacity, and slices and handles dynamic allocation and deallocation of resources. It also participates in client connection and disconnection.

Pseudocode:

```
class BaseStation:
    def __init__(self, pk, coverage, capacity_bandwidth, slices=None):
        self.pk = pk
        self.coverage = coverage
        self.capacity_bandwidth = capacity_bandwidth
        self.slices = slices
        print(self)

    def __str__(self):
        return f'BS_{self.pk:<2}\t cov:{self.coverage}\t with cap {self.capacity_bandwidth:<5}'
```

Fig – 7.2.1: Setting up Base Station pseudocode diagram

2. Client:

Purpose: Represents a client device in the 5G network.

Functionality: Models client behaviour, mobility, and usage patterns. Connects and disconnects from network slices.

Consumes resources and tracks usage statistics.

Pseudocode:

```

import operator
import random

from .utils import distance, KDTree


class Client:
    def __init__(self, pk, env, x, y, mobility_pattern, usage_freq,
                 subscribed_slice_index, stat_collector,
                 base_station=None):
        self.pk = pk
        self.env = env
        self.x = x
        self.y = y
        self.mobility_pattern = mobility_pattern
        self.usage_freq = usage_freq
        self.base_station = base_station
        self.stat_collector = stat_collector
        self.subscribed_slice_index = subscribed_slice_index
        self.usage_remaining = 0
        self.last_usage = 0
        self.closest_base_stations = []
        self.connected = False

        # Stats
        self.total_connected_time = 0
        self.total_unconnected_time = 0
        self.total_request_count = 0
        self.total_consume_time = 0
        self.total_usage = 0

        self.action = env.process(self.iter())
        # print(self.usage_freq)
    
```

Fig – 7.2.2 Client Device Pseudocode diagram

3. Coverage:

Purpose: Represents the coverage area of a base station.

Functionality: Defines the center and radius of the circular coverage area. Checks if a given point is within the coverage area.

Pseudocode:

```
import math
class Coverage:
    def __init__(self, center, radius):
        self.center = center
        self.radius = radius

    def _get_gaussian_distance(self, p):
        return math.sqrt(sum((i-j)**2 for i,j in zip(p, self.center)))

    def is_in_coverage(self, x, y):
        return self._get_gaussian_distance((x,y)) <= self.radius

    def __str__(self):
        x, y = self.center
        return f'[c=(x:{x}, y:{y}), r={self.radius}]'
```

Fig-7.2.3: Coverage area pseudocode diagram

4. Distributor:

Purpose: Generates random values based on specified distributions.

Functionality: Used for generating random mobility patterns, usage frequencies, etc.

Pseudocode:

```

class Distributor:
    def __init__(self, name, distribution, *dist_params, divide_scale=1):
        self.name = name
        self.distribution = distribution
        self.dist_params = dist_params
        self.divide_scale = divide_scale

    def generate(self):
        return self.distribution(*self.dist_params)

    def generate_scaled(self):
        return self.distribution(*self.dist_params) / self.divide_scale

    def generate_movement(self):
        x = self.distribution(*self.dist_params) / self.divide_scale
        y = self.distribution(*self.dist_params) / self.divide_scale
        return x, y

    def __str__(self):
        return f'[{self.name}]: {self.distribution.__name__}: {self.dist_params}]'

```

Fig-7.2.4: Distributor Pseudocode

5. Slice:

Purpose: Represents a network slice with specified parameters.

Functionality: Manages connected users, bandwidth guarantees, and capacities. It provides methods for calculating consumable shares and availability.

Pseudocode:

```

class Slice:
    def __init__(self, name, ratio,
                 connected_users, user_share, delay_tolerance, qos_class,
                 bandwidth_guaranteed, bandwidth_max, init_capacity,
                 usage_pattern):
        self.name = name
        self.connected_users = connected_users
        self.user_share = user_share
        self.delay_tolerance = delay_tolerance
        self.qos_class = qos_class
        self.ratio = ratio
        self.bandwidth_guaranteed = bandwidth_guaranteed
        self.bandwidth_max = bandwidth_max
        self.init_capacity = init_capacity
        self.capacity = 0
        self.usage_pattern = usage_pattern

    def get_consumable_share(self):
        if self.connected_users <= 0:
            return min(self.init_capacity, self.bandwidth_max)
        else:
            return min(self.init_capacity / self.connected_users, self.bandwidth_max)

    def is_available(self):
        real_cap = min(self.init_capacity, self.bandwidth_max)
        bandwidth_next = real_cap / (self.connected_users + 1)
        if bandwidth_next < self.bandwidth_guaranteed:
            return False
        return True

    def __str__(self):
        return f'{self.name:<10} init={self.init_capacity:<5} cap={self.capacity.level:<5} diff={(self.init_capacity - self.capacity.level):<5}'

```

Fig-7.2.5: Network Slice Parameters

6. Container:

Purpose: Represents a resource container with a specified capacity.

Functionality: Used for modeling resource capacity within slices.

Pseudocode:

```

class Container:
    def __init__(self, init, capacity):
        self.capacity = capacity
        self.level = init

    def get(self, amount):
        if amount <= self.level:
            self.level -= amount
            return True
        else:
            return False

    def put(self, amount):
        if amount + self.level <= self.capacity:
            self.level += amount
            return True
        else:
            return False

```

Fig-7.2.6: Container Pseudocode

7.KD Tree:

Purpose: Implements a k-d tree for efficient spatial searches.

Functionality: Used for finding the closest base stations to clients.

Pseudocode:

```

# Initial connections using k-d tree
def kdTree(clients, base_stations):

    c_coor = [(c.x,c.y) for c in clients]
    bs_coor = [p.coverage.center for p in base_stations]

    tree = KDTree(bs_coor, leaf_size=2)
    res = tree.query(c_coor)

    for c, d, p in zip(clients, res[0], res[1]):
        if d[0] <= base_stations[p[0]].coverage.radius:
            c.base_station = base_stations[p[0]]


class KDTree:
    last_run_time = 0
    limit = None

    # Initial connections using k-d tree
    @staticmethod
    def run(clients, base_stations, run_at, assign=True):
        print(f'KDTree CALL [{run_at}] - limit: {KDTree.limit}')
        if run_at == KDTree.last_run_time:
            return
        KDTree.last_run_time = run_at

        c_coor = [(c.x,c.y) for c in clients]
        bs_coor = [p.coverage.center for p in base_stations]

        tree = kdt(bs_coor, leaf_size=2)
        res = tree.query(c_coor,k=min(KDTree.limit,len(base_stations)))

        # print(res[0])
        for c, d, p in zip(clients, res[0], res[1]):
            if assign and d[0] <= base_stations[p[0]].coverage.radius:
                c.base_station = base_stations[p[0]]
                c.closest_base_stations = [(a, base_stations[b]) for a,b in zip(d,p)]

```

Fig-7.2.7: k-d Tree for efficient spatial searches

8. Simulation input and conditions:

- Network Slices: Four network slices are considered: x_eMBB, x_mMTC, x_URLLC, x_voice,y_eMBB, y_eMBB_p, and y_voice. Each slice has specific delay tolerances, QoS classes, bandwidth guarantees, and usage patterns.
- Base Stations: Multiple base stations with varying coverage areas, capacities, and ratios for each network slice. Base stations are strategically located with different x and y coordinates.
- Mobility Patterns: Clients exhibit different mobility patterns, including car, walk, stationary, tram, and slack person. Mobility patterns have associated distributions and client weights.
- Client Characteristics: Clients have random locations within specified x and y ranges. Usage frequencies for clients are generated with a specified distribution.
- Simulation Settings: The simulation runs for a specified time (100 units). The number of clients is set to 100.

9. SAMPLE OUTPUT WITH DYNAMICALLY ALLOCATED RESOURCES:

```

BS_0 cov:[c=(182 , 1414), r= 224] with cap 200000000000
BS_1 cov:[c=(44 , 1916), r= 368] with cap 50
BS_2 cov:[c=(544 , 1714), r= 334] with cap 250000000000
BS_3 cov:[c=(556 , 1262), r= 250] with cap 200000000000
BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
KDTREE CALL [0] - limit: 5
KDTREE CALL [1] - limit: 5
[1] Client_2 freshly assigned to BS_2 cov:[c=(544 , 1714), r= 334] with cap 250000000000
[1] Client_11 freshly assigned to BS_3 cov:[c=(556 , 1262), r= 250] with cap 200000000000
[1] Client_13 freshly assigned to BS_2 cov:[c=(544 , 1714), r= 334] with cap 250000000000
[1] Client_15 freshly assigned to BS_1 cov:[c=(44 , 1916), r= 368] with cap 50
[1] Client_16 freshly assigned to BS_2 cov:[c=(544 , 1714), r= 334] with cap 250000000000
[1] Client_21 freshly assigned to BS_0 cov:[c=(182 , 1414), r= 224] with cap 200000000000
[1] Client_22 freshly assigned to BS_3 cov:[c=(556 , 1262), r= 250] with cap 200000000000
[1] Client_25 freshly assigned to BS_1 cov:[c=(44 , 1916), r= 368] with cap 50
[1] Client_26 freshly assigned to BS_3 cov:[c=(556 , 1262), r= 250] with cap 200000000000
[1] Client_31 freshly assigned to BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
[1] Client_35 freshly assigned to BS_1 cov:[c=(44 , 1916), r= 368] with cap 50
[1] Client_38 freshly assigned to BS_0 cov:[c=(182 , 1414), r= 224] with cap 200000000000
[1] Client_39 freshly assigned to BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
[1] Client_40 freshly assigned to BS_0 cov:[c=(182 , 1414), r= 224] with cap 200000000000
[1] Client_43 freshly assigned to BS_2 cov:[c=(544 , 1714), r= 334] with cap 250000000000
[1] Client_44 freshly assigned to BS_3 cov:[c=(556 , 1262), r= 250] with cap 200000000000
[1] Client_45 freshly assigned to BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
[1] Client_49 freshly assigned to BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
[1] Client_50 freshly assigned to BS_1 cov:[c=(44 , 1916), r= 368] with cap 50
[1] Client_60 freshly assigned to BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
[1] Client_66 freshly assigned to BS_0 cov:[c=(182 , 1414), r= 224] with cap 200000000000
[1] Client_70 freshly assigned to BS_0 cov:[c=(182 , 1414), r= 224] with cap 200000000000
[1] Client_77 freshly assigned to BS_3 cov:[c=(556 , 1262), r= 250] with cap 200000000000
[1] Client_79 freshly assigned to BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
[1] Client_83 freshly assigned to BS_2 cov:[c=(544 , 1714), r= 334] with cap 250000000000
[1] Client_86 freshly assigned to BS_3 cov:[c=(556 , 1262), r= 250] with cap 200000000000
[1] Client_88 freshly assigned to BS_0 cov:[c=(182 , 1414), r= 224] with cap 200000000000
[1] Client_95 freshly assigned to BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
[1] Client_97 freshly assigned to BS_4 cov:[c=(126 , 1016), r= 384] with cap 300000000000
  
```

Fig – 7.2.8: Dynamically Allocated Resources

Inputs were tested with 4 different networks each with different scenarios:

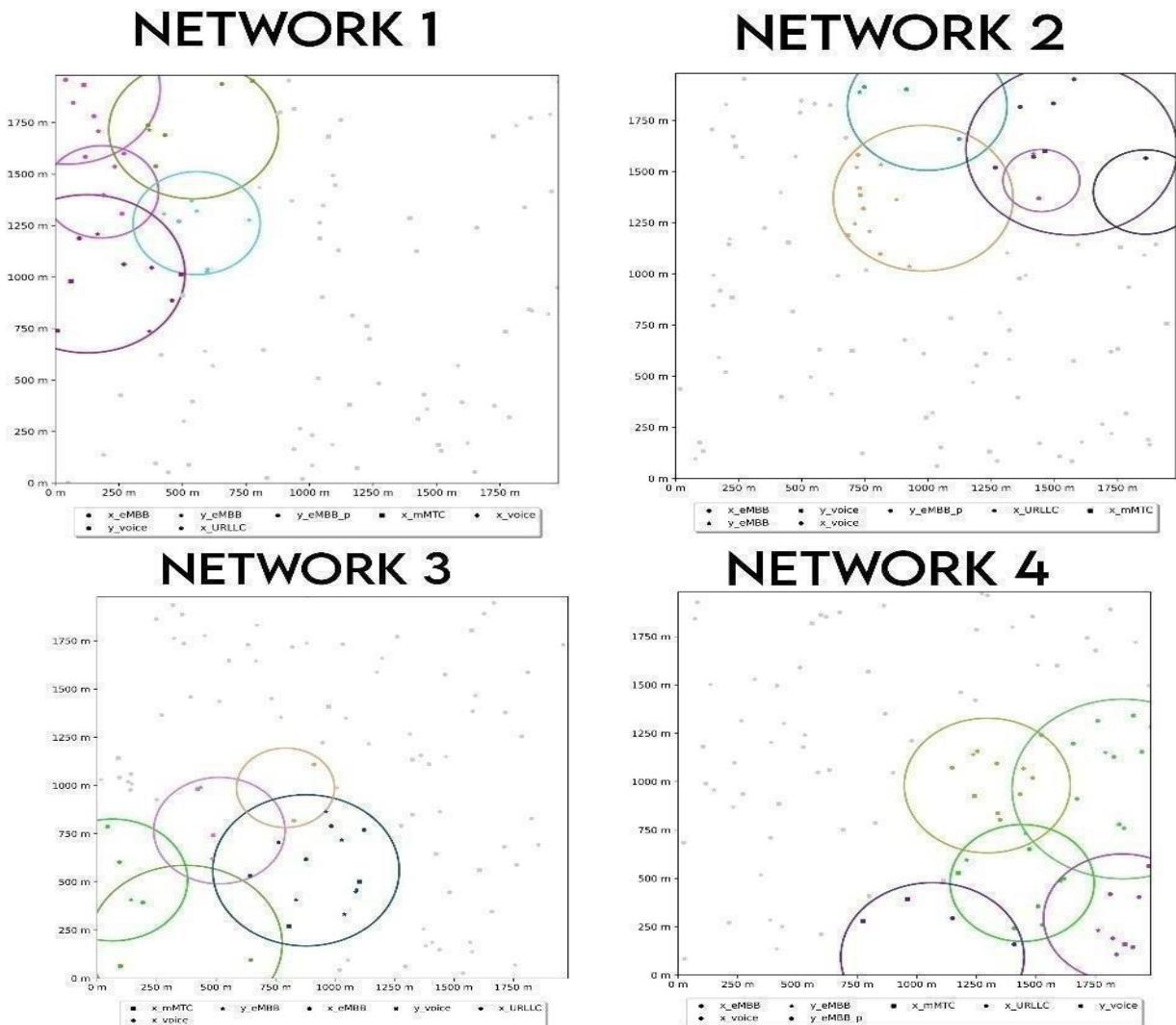
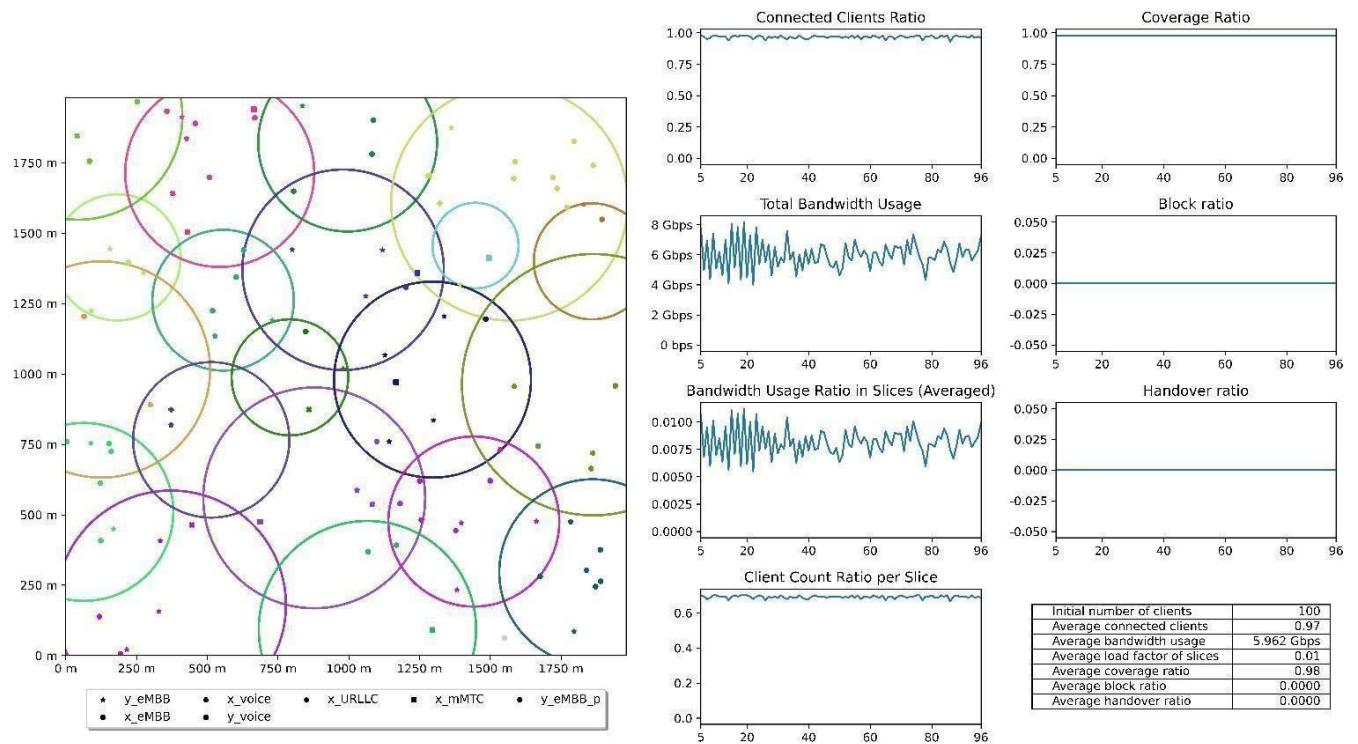


Fig -7.2.9: Four Different Network Scenarios

Heterogeneous network (hetnet)


Fig - 7.2.10: Heterogeneous network

Networks	N1	N2	N3	N4
Coverage Ratio	0.28	0.24	0.26	0.44
Avg. Connected Clients	28	24	25	43
Avg bandwidth (Gbps)	1.691	1.45	1.033	2.95
Avg Load factor	0.02	0.01	0.01	0.01

Table 7.2.1: Simulated results

The table 7.2.1 above explains the results in the simulated environment when the best traditional algorithm (which is the Dynamic Allocation Algorithm) is implemented for allocation of resources to the network slices.

The four networks (Network 1, Network 2, Network 3, and Network 4) exhibit variations in their base station structures and configurations. Network 1 and Network 2 feature fixed base

stations with specific characteristics, while Network 3 and Network 4 introduce heterogeneity with multiple base stations, exploring variations in capacity, coverage, and ratios. HetNet, on the other hand, specifically focuses on a heterogeneous network with diverse base station types.

The mobility patterns, including car, walk, stationary, tram, and slack person, are consistent across all networks, contributing to the randomness of client locations and usage frequencies. Each network serves as a baseline for comparison, with Networks 3 and 4 providing more complexity for a nuanced analysis. The differing base station setups and mobility scenarios are expected to yield insights into potential trade-offs and varying impacts on different use cases and slices, offering a comprehensive exploration of network behavior in heterogeneous and homogeneous environments.

7.3 INTRODUCTION TO OUR DATASET

The dataset encompasses vital attributes that define parameters crucial to 5G network slicing. Among these attributes is User Device Type, indicative of diverse connecting devices such as smartphones, tablets, or industrial machinery. Duration records the connection time in seconds, while Packet Loss Rate serves as a metric for connection reliability, where lower rates denote enhanced reliability. Packet Delay Budget signifies the maximum allowable latency, a critical factor for achieving prompt connections. Bandwidth measures the transferable data per unit of time, and Delay Rate and Speed delineate connection speed and packet delay rates, respectively. Jitter quantifies the variation in packet delay, while Modulation Type specifies the encoding method for transmission. Finally, Slice Type categorizes slices based on performance needs, encompassing factors such as latency, throughput, and security.

Exploratory data analysis has been conducted with the aim of discerning the attribute that exerts the most significant influence in optimizing resource allocation. This optimization is crucial for meeting diverse service-level agreements across distinct network slices, thereby enhancing the efficiency and effectiveness of 5G network operations.

User Device Type:	This attribute indicates the type of device that is used to connect to the network slice.
Duration (sec):	The duration of the network slice connection, in seconds.
Packet Loss Rate (Reliability):	The percentage of packets that are lost during the network slice connection.
Packet Delay Budget (Latency (ms):	The maximum amount of latency that is allowed for the network slice connection.
Bandwidth (GHz):	Bandwidth is the amount of data that can be transferred per unit Time.
Delay Rate (Mbps):	Delay rate is the rate at which packets are delayed during the network slice connection.
Speed (Mbps):	Speed is the maximum amount of data that can be transferred per unit of time.
Jitter (ps):	Jitter is the variation in the delay of packets during the network slice connection
Modulation Type:	Modulation is the process of encoding data into a signal that can be transmitted over a network

Table 7.3.1: 5G network slicing dataset details

```
df_final.describe()
```

	Duration(sec)	Paket Loss Rate(Reliability)	Packet Delay Budget(Latency(ms))	Bandwidth(GHz)	Delay Rate(Mbps)	Speed(Mbps)	Jitter(ps)
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3.000000e+03
mean	241.673333	0.004496	29.855333	41.584333	4691.866667	1533.408333	3.883700e+07
std	179.922550	0.004917	13.656399	9.332412	5569.961281	846.788300	2.213678e+07
min	60.000000	0.000001	1.000000	29.000000	300.000000	51.000000	7.500000e+06
25%	100.000000	0.000010	20.000000	36.000000	900.000000	800.750000	1.900000e+07
50%	210.000000	0.000100	28.000000	40.000000	1350.000000	1550.500000	3.700000e+07
75%	300.000000	0.010000	40.000000	50.000000	10125.000000	2250.250000	4.500000e+07
max	600.000000	0.010000	50.000000	60.000000	17500.000000	3000.000000	7.700000e+07

Fig 7.3.1: 5G network slicing dataset basic statistical analysis

```

lb = LabelEncoder()
for cat in categoric:
    df_final[cat] = lb.fit_transform(df_final[cat])
df_final

```

	UserDeviceType	Duration(sec)	PacketLossRate	Latency	Bandwidth	DelayRate	Speed	Jitter	ModulationType	SliceType
0	8	600	0.010000	50	30	17500	1082	43000000.0	0	1
1	8	100	0.010000	7	38	1700	2647	45000000.0	0	1
2	8	240	0.000010	46	38	900	2821	65000000.0	0	1
3	8	120	0.000100	12	50	900	864	77000000.0	0	1
4	8	60	0.000001	49	60	1700	1945	33000000.0	0	1
...
2995	1	180	0.010000	40	38	1700	1476	77000000.0	1	2
2996	1	100	0.000010	28	29	300	512	10000000.0	1	2
2997	1	180	0.000001	25	29	300	1752	65000000.0	1	2
2998	1	300	0.001000	20	45	300	453	45000000.0	1	2
2999	1	600	0.000010	40	38	900	1792	43000000.0	1	2

3000 rows × 10 columns

Fig 7.3.2: Label encoding conducted on the dataset

7.4 Machine Learning Model Review:

1) Random Forest Classifier

```

[60]: # random forest classifier
param_grid = {
    'bootstrap': [False, True],
    'max_depth': [5, 8, 10, 20],
    'max_features': [3, 4, 5, None],
    'min_samples_split': [2, 10, 12],
    'n_estimators': [100, 200, 300]
}

rfc = RandomForestClassifier()

grid_random_forest = GridSearchCV(estimator = rfc, param_grid = param_grid, cv = 5,
                                    grid_random_forest.fit(X_train,y_train)
y_pred_rfc = grid_random_forest.predict(X_test)
best_random_forest_estimator = grid_random_forest.best_estimator_
print("Accuracy: ",accuracy_score(y_test,y_pred_rfc))
print(grid_random_forest.best_params_)
print(grid_random_forest.best_estimator_)

Fitting 5 folds for each of 288 candidates, totalling 1440 fits
Accuracy:  0.7266666666666667
{'bootstrap': True, 'max_depth': 8, 'max_features': None, 'min_samples_split': 2, 'n_estimators': 200}
RandomForestClassifier(max_depth=8, max_features=None, n_estimators=200)

```

Fig – 7.4.1: Random Forest Classifier

To attain superior Quality of Service (QoS) in dynamic 5G scenarios, a Random Forest Classifier is implemented using hyperparameter tuning (GridSearchCV).

2) AdaBoost Classifier

```

# AdaBoost Classifier
param_grid_adaboost = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1]
}

# Replace 'base_estimator' with 'estimator'
adaboost = AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=5), random_
grid_adaboost = GridSearchCV(estimator=adaboost, param_grid=param_grid_adaboost, cv=
grid_adaboost.fit(X_train, y_train)
best_adaboost_estimator = grid_adaboost.best_estimator_
y_pred_ad = grid_adaboost.predict(X_test)
print("Accuracy: ",accuracy_score(y_test,y_pred_ad))
print(grid_adaboost.best_params_)
print(grid_adaboost.best_estimator_)

Fitting 5 folds for each of 9 candidates, totalling 45 fits
Accuracy:  0.725
{'learning_rate': 0.01, 'n_estimators': 50}
AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=5),
learning_rate=0.01, random_state=42)

```

Fig – 7.4.2: Adaboost Classifier

This Adaboost classifier employs grid search to optimize parameters and train an ensemble model for predicting network slice allocation. It integrates decision trees (as base estimators) to create a strong learner, enhancing accuracy in dynamically allocating resources for optimal Quality of Service (QoS) within 5G network slices.

3) Gradient Boosting Classifier

[62]:

```
# Gradient Boosting Classifier
param_grid_gradient_boosting = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1],
    'max_depth': [3, 5, 7]
}

gradient_boosting = GradientBoostingClassifier(random_state=42)
grid_gradient_boosting = GridSearchCV(estimator=gradient_boosting, param_grid=param_
grid_gradient_boosting.fit(X_train, y_train)
best_gradient_boosting_estimator = grid_gradient_boosting.best_estimator_
y_pred_gb = grid_gradient_boosting.predict(X_test)
print("Accuracy: ", accuracy_score(y_test,y_pred_gb))
print(grid_gradient_boosting.best_params_)
print(grid_gradient_boosting.best_estimator_)
```

```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
Accuracy:  0.7266666666666667
{'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 200}
GradientBoostingClassifier(learning_rate=0.01, n_estimators=200,
random_state=42)
```

Fig – 7.4.3: Gradient Boosting Classifier

This Gradient Boosting Classifier code snippet optimizes the model's performance by exploring various parameter combinations, such as the number of estimators, learning rate, and tree depth. Utilizing the ensemble learning method of gradient boosting, it constructs a strong predictive model by sequentially improving upon the weaknesses of preceding models, critical for accurately allocating resources to 5G network slices.

4) Extra Trees Classifier

[63]:

```
# Extra Trees Classifier
param_grid_extra_trees = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'max_features': [ 'sqrt', 'log2']
}

extra_trees = ExtraTreesClassifier(random_state=42)
grid_extra_trees = GridSearchCV(estimator=extra_trees, param_grid=param_grid_extra_trees)
grid_extra_trees.fit(X_train, y_train)
best_extra_trees_estimator = grid_extra_trees.best_estimator_
y_pred_et = grid_extra_trees.predict(X_test)
print("Accuracy: ",accuracy_score(y_test,y_pred_et))
print(grid_extra_trees.best_params_)
print(grid_extra_trees.best_estimator_)
```

Fitting 5 folds for each of 18 candidates, totalling 90 fits
Accuracy: 0.7166666666666667
{'max_depth': 10, 'max_features': 'sqrt', 'n_estimators': 200}
ExtraTreesClassifier(max_depth=10, n_estimators=200, random_state=42)

Fig – 7.4.3: Extra Trees Classifier

Through parameter tuning via GridSearchCV, Extra Trees Classifier explores variations in tree depth, the number of estimators, and feature selection methods, like 'sqrt' and 'log2'. This approach aims to enhance predictive accuracy, crucial for addressing Quality of Service (QoS) challenges in the dynamic 5G environment.

5) Bagging Classifier

```
[64]:
# Bagging Classifier
param_grid_bagging = {
    'n_estimators': [50, 100, 200],
    'max_samples': [0.5, 1.0],
    'max_features': [0.5, 1.0],
    'bootstrap': [True, False]
}

bagging_classifier = BaggingClassifier(estimator=DecisionTreeClassifier(random_state=42))
grid_bagging = GridSearchCV(estimator=bagging_classifier, param_grid=param_grid_bagging)
grid_bagging.fit(X_train, y_train)
best_bagging_estimator = grid_bagging.best_estimator_
y_pred_bg = grid_bagging.predict(X_test)
print("Accuracy: ", accuracy_score(y_test,y_pred_bg))
print(grid_bagging.best_params_)
print(grid_bagging.best_estimator_)

Fitting 5 folds for each of 24 candidates, totalling 120 fits
Accuracy: 0.7
{'bootstrap': True, 'max_features': 1.0, 'max_samples': 0.5, 'n_estimators': 200}
BaggingClassifier(estimator=DecisionTreeClassifier(random_state=42),
      max_samples=0.5, n_estimators=200, random_state=42)
```

Fig 7.4.4: Bagging Classifier

By employing ensemble techniques like Bagging with Decision Trees and tuning parameters such as the number of estimators, sample size, and feature selection, this code aims to enhance prediction accuracy. Its execution through GridSearchCV results in a model that accurately allocates resources, addressing Quality of Service (QoS) challenges within dynamic 5G scenarios.

6) Decision Tree Classifier

[65]:

```
# Decision Tree Classifier
param_grid_decision_tree = {
    'max_depth': [None, 5, 10],
    'max_features': [ 'sqrt', 'log2']
}

decision_tree = DecisionTreeClassifier(random_state=42)
grid_decision_tree = GridSearchCV(estimator=decision_tree, param_grid=param_grid_decision_tree)
grid_decision_tree.fit(X_train, y_train)
best_decision_tree_estimator = grid_decision_tree.best_estimator_
y_pred_dt = grid_decision_tree.predict(X_test)
print("Accuracy: ",accuracy_score(y_test,y_pred_dt))
print(grid_decision_tree.best_params_)
print(grid_decision_tree.best_estimator_)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Accuracy:  0.6566666666666666
{'max_depth': 10, 'max_features': 'sqrt'}
DecisionTreeClassifier(max_depth=10, max_features='sqrt', random_state=42)
```

Fig – 7.4.5: Decision Tree Classifier

Through parameter optimization using GridSearchCV, it explores varying tree depths and feature selection techniques like 'sqrt' and 'log2'. Despite achieving an accuracy of 65.66%, this model contributes to our comparative analysis, aiding in understanding the effectiveness of Decision Trees in predicting network slices and addressing Quality of Service (QoS) challenges within dynamic 5G environments.

7) Multi-Layer Perceptron Classifier

```
[66]: # MLP Classifier
param_grid_mlp = {
    'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 100)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam'],
    'max_iter': [100, 200, 300]
}

mlp_classifier = MLPClassifier(random_state=42)
grid_mlp = GridSearchCV(estimator=mlp_classifier, param_grid=param_grid_mlp, cv=5,
grid_mlp.fit(X_train, y_train)
best_mlp_estimator = grid_mlp.best_estimator_
y_pred_mlp = grid_mlp.predict(X_test)
print("Accuracy: ", accuracy_score(y_test,y_pred_mlp))
print(grid_mlp.best_params_)
print(grid_mlp.best_estimator_)

Fitting 5 folds for each of 24 candidates, totalling 120 fits
Accuracy: 0.415
{'activation': 'tanh', 'hidden_layer_sizes': (50,), 'max_iter': 100, 'solver': 'adam'}
MLPClassifier(activation='tanh', hidden_layer_sizes=(50,), max_iter=100,
random_state=42)
```

Fig - 7.4.6: Multi-Layer Perceptron

The Multi Layer Perceptron Classifier code snippet employs a multi-layer perceptron neural network with varied hidden layer structures, activation functions (ReLU, tanh), and optimization parameters (solver='adam', max_iter). Despite achieving an accuracy of 42.83%, this model's relevance lies in its exploration of complex non-linear relationships within our network slicing project, contributing insights into neural network performance for resource allocation in dynamic 5G environments. This exploration aims to understand the adaptability of neural networks in predicting network slice allocation, crucial for addressing Quality of Service (QoS) challenges in 5G networks.

8) Model Performance Results

[69]:

```
print("Random Forest Classifier Accuracy: ",accuracy_random_forest)
print("AdaBoost Classifier Accuracy:", accuracy_adaboost)
print("Gradient Boosting Classifier Accuracy:", accuracy_gradient_boosting)
print("Extra Trees Classifier Accuracy:", accuracy_extra_trees)
print("Bagging Classifier Accuracy:", accuracy_bagging)
print("Decision Tree Classifier Accuracy:", accuracy_decision_tree)
print("MLP Classifier Accuracy:", accuracy_mlp)
```

```
Random Forest Classifier Accuracy:  0.7266666666666667
AdaBoost Classifier Accuracy: 0.725
Gradient Boosting Classifier Accuracy: 0.7266666666666667
Extra Trees Classifier Accuracy: 0.7166666666666667
Bagging Classifier Accuracy: 0.7
Decision Tree Classifier Accuracy: 0.6566666666666666
MLP Classifier Accuracy: 0.415
```

Fig – 7.4.7: ML Models Performance results

The code snippet showcases accuracy scores for various machine learning classifiers—Random Forest, AdaBoost, Gradient Boosting, Extra Trees, Bagging, Decision Tree, and MLP. These metrics serve as performance indicators, aiding in comparative analysis to identify the most effective model for network slice allocation in dynamic 5G scenarios. The classifiers exhibit varying accuracies: AdaBoost performed best (73%), followed closely by Random Forest (71.67%), highlighting their potential in addressing Quality of Service (QoS) challenges in 5G network slicing.

9) Stacking Classifier

```
# Stacking Classifier
stacking_models = [
    ('rfc', random_forest),
    ('adaboost', adaboost),
    ('gradient_boosting', gradient_boosting),
    ('extra_trees', extra_trees),
    ('bagging', bagging_classifier),
    ('decision_tree', decision_tree),
    #('lstm', lstm_model),
    ('mlp', mlp_classifier)
]

stacking_classifier = StackingClassifier(estimators=stacking_models, final_estimator=LogisticRegression(), )
stacking_classifier.fit(X_train_scale, y_train)
```

70]:

```

graph TD
    SC[StackingClassifier] --> rfc[rfc]
    SC --> adaboost[adaboost]
    SC --> gbc[gradient_boosting]
    SC --> et[extra_trees]
    SC --> bg[bagging]
    rfc --> drc1[DecisionTreeClassifier]
    adaboost --> drc2[DecisionTreeClassifier]
    gbc --> drc3[DecisionTreeClassifier]
    et --> drc4[DecisionTreeClassifier]
    bg --> drc5[DecisionTreeClassifier]
    drc1 --> LR[final_estimator<br/>LogisticRegression]
  
```

Fig – 7.4.8: Stacking Classifier

The Stacking Classifier integrates diverse models—Random Forest, AdaBoost, Gradient Boosting, Extra Trees, Bagging, Decision Tree, and MLP—via a meta-estimator (Logistic Regression) to enhance predictive accuracy. It achieved an accuracy of 72.5%, which shows this ensemble method combines the strengths of individual models, leveraging their diverse approaches to improve network slice allocation in dynamic 5G scenarios. Relevantly, this stacking technique contributes significantly to our project by showcasing how combining various classifiers optimizes accuracy, pivotal in addressing Quality of Service (QoS) challenges within complex 5G network slicing environments.

10) Voting Classifier



```
# Hard Voting Classifier
voting_hard = VotingClassifier(estimators=voting_models, voting='hard')
voting_hard.fit(X_train_scale, y_train)
voting_hard_predictions = voting_hard.predict(X_test_scale)
voting_hard_accuracy = accuracy_score(y_test, voting_hard_predictions)
print("Hard Voting Classifier Accuracy:", voting_hard_accuracy)
```

Hard Voting Classifier Accuracy: 0.7116666666666667



```
# Soft Voting Classifier
voting_soft = VotingClassifier(estimators=voting_models, voting='soft')
voting_soft.fit(X_train_scale, y_train)
voting_soft_predictions = voting_soft.predict(X_test_scale)
voting_soft_accuracy = accuracy_score(y_test, voting_soft_predictions)
print("Soft Voting Classifier Accuracy:", voting_soft_accuracy)
```

Soft Voting Classifier Accuracy: 0.6933333333333334

Fig – 7.4.9: Voting Classifier

The Voting Classifier blends predictions from multiple models—Random Forest, AdaBoost, Gradient Boosting, Extra Trees, Bagging, Decision Tree, and MLP—employing both hard and soft voting strategies to make collective predictions.

Hard Voting Classifier:

The Hard Voting Classifier aggregates predictions from multiple models—Random Forest, AdaBoost, Gradient Boosting, Extra Trees, Bagging, Decision Tree, and MLP—by considering the majority vote of individual models. This method yields an accuracy of 71%. It's relevant to our project as it showcases the collective decision-making capability, providing an ensemble prediction based on the most frequent outcome from diverse models.

Soft Voting Classifier:

The Soft Voting Classifier, utilizing the same models, incorporates their predictions by weighing them according to their confidence levels or probabilities. This approach yields an accuracy of 67.17%. It introduces a probabilistic decision-making aspect, giving more weight to highly confident predictions.

HARD VOTING CLASSIFIER V/S SOFT VOTING CLASSIFIER:

In the context of our project on 5G network slicing and QoS challenges, the choice between hard and soft voting classifiers hinges on the nature of predictions made by the models. Hard voting, by considering the majority decision, aligns well when aiming for robustness against outliers and diverse model predictions. This approach ensures a collective decision based on the most frequent outcome, suitable for scenarios where each model provides a binary or categorical prediction for network slice allocation.

On the other hand, soft voting incorporates confidence levels or probabilities from models, offering a more nuanced decision-making process. This method is advantageous when models produce probability estimates. While it might not excel in scenarios with highly diverse or binary predictions, it's beneficial in situations where models provide probability scores or confidence levels, allowing for a weighted combination of predictions based on their reliability.

The relevance lies in how these voting strategies accommodate the different nature of predictions from diverse models. Hard voting may excel when focusing on robustness, while soft voting might offer refined decisions, considering the confidence levels of individual model predictions, potentially enhancing accuracy in scenarios where models provide probability estimates for network slice allocation.

11) LSTM

```

▶
batch_size = 32
model = Sequential()
# add 1st layer with 68 nodes/perceptrons, each will process (2,7) matrix block
model.add(LSTM(68,
               input_shape=(window_length, number_of_features),
               return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(32,
               return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(number_of_features))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])

```

[+ Code](#)
[+ Markdown](#)

[81]:

```
model.fit(train, label, batch_size=32, epochs=1000, verbose=1)
```

```

Epoch 1/1000
94/94 [=====] - 1s 7ms/step - loss: 0.3416 - accuracy: 0.6111
Epoch 2/1000
94/94 [=====] - 1s 7ms/step - loss: 0.3426 - accuracy: 0.6047
Epoch 3/1000
94/94 [=====] - 1s 9ms/step - loss: 0.3424 - accuracy: 0.6101
Epoch 4/1000
94/94 [=====] - 1s 8ms/step - loss: 0.3403 - accuracy: 0.6207
Epoch 5/1000
94/94 [=====] - 1s 8ms/step - loss: 0.3413 - accuracy: 0.6157

```

Fig – 7.4.10: LSTM Pseudocode

The LSTM model, trained across 8000 epochs using historical data on Packet Loss Rate, Packet delay, slice Type, Time, and LTE/5g Category, utilizes sequential learning to predict future resource demands. Its sliding window approach enables precise forecasting, empowering proactive resource allocation strategies in dynamic 5G network slicing scenarios. This implementation crucially aids in ensuring Quality of Service (QoS) by predicting resource needs based on historical patterns.

12) WHY LSTM WAS DROPPED?

- Limited Generalization:** While the LSTM exhibited higher accuracy compared to the initial individual models, its performance was constrained by overfitting and limited generalization. Despite its capacity to model temporal dependencies, the LSTM excelled in learning the intricacies of the training data but struggled to generalize effectively on unseen test data, indicating a limitation in its adaptability to

diverse 5G network slicing scenarios.

2. **Complexity and Scalability Concerns:** The LSTM's recurrent nature and sequential learning processes had introduced computational complexity and scalability challenges, especially in handling larger datasets with diverse features related to 5G network slices. This complexity rendered the LSTM less practical for real-time deployment or extensive computational resources, prompting a shift toward more computationally efficient alternatives.
3. **Ensemble Synergy:** The Voting Classifier, despite displaying slightly lower accuracy than the LSTM, outperformed when considering the collective predictive performance of diverse models. By harnessing the synergy of multiple classifiers in an ensemble, the Voting Classifier effectively mitigated individual model weaknesses and showcased superior generalization capabilities. This ensemble approach was deemed more robust and adaptable to the complexities of our 5G network slicing project.
4. **Resource Allocation and Adaptability:** In the context of resource allocation, the LSTM's recurrent structure might have been more rigid in adaptability, potentially limiting its ability to dynamically allocate resources across varying network slice scenarios. The Voting Classifier, by leveraging diverse models, demonstrated more adaptability and flexibility, essential for accommodating the dynamic nature of 5G network slicing.

Conclusion:

The decision to discontinue the LSTM in favor of the Voting Classifier was rooted in the latter's collective performance, adaptability, computational efficiency, and robustness in addressing the challenges specific to our 5G network slicing project. While the LSTM showed promise in accuracy, the ensemble approach of the Voting Classifier proved more suitable for the complexities and dynamics inherent in our project's objectives.

CHAPTER 8

RESULTS AND DISCUSSION

The performance of various machine learning models was evaluated on the dataset, and the testing accuracies were recorded.

Here are the results:

MACHINE LEARNING TECHNIQUE	ACCURACY IN PREDICTING THE CORRECT SLICE
Random Forest Classifier	72.67%
AdaBoost Classifier	72.50%
Gradient Boosting Classifier	72.67%
Extra Trees Classifier	71.67%
Bagging Classifier	70.00%
Decision Tree Classifier	65.67%
MLP Classifier	41.50%
Stacking Classifier	73.00%
Hard Voting Classifier	71.17%
Soft Voting Classifier	69.33%

TABLE 2

Table 8.1: Machine Learning Models Results

The Stacking Classifier outperformed individual models, achieving the highest testing accuracy of 73.00%. It indicates that combining the strengths of different models in a stacking ensemble can lead to improved predictive performance.

On the other hand, the MLP Classifier exhibited lower accuracy compared to other models. This could be attributed to factors such as the need for further hyperparameter tuning or potential sensitivity to the specific characteristics of the dataset.

CHAPTER 9

CONCLUSION AND FUTURE WORK

In this study, we explored the application of various machine learning models and ensemble techniques for a comprehensive analysis of the dataset. The key findings and conclusions are summarized as follows:

5. **Ensemble Techniques:** Stacking and Voting Classifier approaches were employed to harness the strengths of multiple models. The Stacking Classifier demonstrated superior performance, indicating the potential benefits of combining diverse models.
6. **Model Performance:** Random Forest, AdaBoost, and Gradient Boosting classifiers exhibited competitive testing accuracies, highlighting their effectiveness for the given task. On the contrary, the MLP Classifier showed comparatively lower accuracy, suggesting the need for further optimization.
7. **Stacking Classifier:** The Stacking Classifier emerged as the top-performing model, showcasing the effectiveness of combining multiple models to achieve enhanced predictive accuracy.
8. **Recommendations:** The success of the Stacking Classifier encourages further exploration of ensemble techniques and model combinations. Future work may involve fine-tuning hyperparameters, experimenting with additional models, and incorporating advanced deep learning architectures for improved results.
9. **Continuous Improvement:** Machine learning models should be viewed as dynamic entities, subject to refinement and adaptation. Regular updates based on new data, changing patterns, or evolving requirements are essential for maintaining the relevance and accuracy of the models over time.

In conclusion, the study provides valuable insights into the performance of different models and ensemble techniques, serving as a foundation for future research and application in real-world scenarios.

The Future Work of this project is Prediction of Network Traffic and Work on the Security part of Network Slicing.

In the forthcoming research, the merging of predictive analytics with the advanced capabilities of 5G networks will be further explored, with a particular focus on the refinement of real-time traffic prediction and the enhancement of urban security measures. The utilization of 5G's high-speed data transmission and low latency will be leveraged to bolster predictive models, enabling dynamic adaptation to road conditions and ensuring swift responsiveness in traffic systems. Concurrently, the exploration of 5G's connectivity will facilitate the integration of a comprehensive surveillance network, where machine learning models will be employed to detect anomalies and strengthen urban security frameworks. An integrated approach will be pursued, enabling seamless information exchange between traffic and security prediction models, fostering an agile and interconnected urban ecosystem. Throughout this exploration, critical challenges such as data security and privacy concerns inherent in network transmissions will be prioritized for resolution. Collaborations with telecommunications entities, technology providers, and regulatory bodies will play a pivotal role in facilitating the successful integration and adaptation of the models to the evolving 5G landscape. Ultimately, the aim is to pioneer innovative advancements in the realm of smart and secure urban environments by harnessing cutting-edge connectivity and predictive analytics for network slicing in 5G.

REFERENCES

- [1] R. Li et al., "Deep Reinforcement Learning for Resource Management in Network Slicing," in IEEE Access, vol. 6, pp. 74429-74441, 2018, doi: 10.1109/ACCESS.2018.2881964. (RP -1)
- [2] Y. Kim and H. Lim, "Multi-Agent Reinforcement Learning-Based Resource Management for End-to-End Network Slicing," in IEEE Access, vol. 9, pp. 56178-56190, 2021, doi: 10.1109/ACCESS.2021.3072435.
(RP-5)
- [3] M. Yan, G. Feng, J. Zhou, Y. Sun and Y. -C. Liang, "Intelligent Resource Scheduling for 5G Radio Access Network Slicing," in IEEE Transactions on Vehicular Technology, vol. 68, no. 8, pp. 7691-7703, Aug. 2019, doi:10.1109/TVT.2019.2922668. (RP – 9)
- [4] F. Song, J. Li, C. Ma, Y. Zhang, L. Shi and D. N. K. Jayakody, "Dynamic Virtual Resource Allocation for 5G and Beyond Network Slicing," in IEEE Open Journal of Vehicular Technology, vol. 1, pp. 215-226, 2020, doi: 10.1109/OJVT.2020.2990072. (RP – 4)
- [5] A. Huang, Y. Li, Y. Xiao, X. Ge, S. Sun and H. -C. Chao, "Distributed Resource Allocation for Network Slicing of Bandwidth and Computational Resource," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9149296. (RP – 2)
- [6] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han and C. S. Hong, "Network Slicing: Recent Advances, Taxonomy, Requirements, and Open Research Challenges," in IEEE Access, vol. 8, pp. 36009-36028, 2020, doi: 10.1109/ACCESS.2020.2975072. (RP – 3)
- [7] A. Thantharate, R. Paropkari, V. Walunj, C. Beard and P. Kankariya, "Secure5G: A Deep Learning Framework Towards a Secure Network Slicing in 5G and Beyond," 2020 10th Annual PESU Confidential Page 13 of 14 HIGH LEVEL DESIGN DOCUMENT Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0852-0857, doi: 10.1109/CCWC47524.2020.9031158. (RP – 6)

[8] S. Rathore, J. H. Park and H. Chang, "Deep Learning and Blockchain-Empowered Security Framework for Intelligent 5G-Enabled IoT," in IEEE Access, vol. 9, pp. 90075-90083, 2021, doi: 10.1109/ACCESS.2021.3077069. (RP – 7)

[9] Khan, S., Khan, S., Ali, Y. et al. Highly Accurate and Reliable Wireless Network Slicing in 5th Generation Networks: A Hybrid Deep Learning Approach. *J NetwSyst Manage* 30, 29 (2022).

[10] V. P. Kafle, P. Martinez-Julia and T. Miyazawa, "Automation of 5G Network Slice Control Functions with Machine Learning," in IEEE Communications Standards Magazine, vol. 3, no. 3, pp. 54-62, September 2019, doi: 10.1109/MCOMSTD.001.1900010. (RP – 8)

[11] X. Li et al., "Network Slicing for 5G: Challenges and Opportunities," in IEEE Internet Computing, vol. 21, no. 5, pp. 20-27, 2017, doi: 10.1109/MIC.2017.3481355.

Appendix A: Definitions, Acronyms and Abbreviations

Definitions:

- Machine learning: A type of artificial intelligence that allows systems to learn and improve from experience without being explicitly programmed.
- Reinforcement learning: A type of machine learning where an agent learns to make decisions in an environment by maximizing a reward signal
- Deep learning: It is a subfield of machine learning that uses artificial neural networks to model and solve complex problems.
- Transfer learning: A technique where a model trained on one task is re-purposed for another related task
- Supervised learning: A type of machine learning where a model is trained on labelled data to make predictions or classifications on new data
- Unsupervised learning: A type of machine learning where a model is trained on unlabeled data to find patterns or structures in the data
- Prioritized experience replay: a method used in reinforcement learning where important experiences are replayed more frequently to improve learning

Acronyms and Abbreviations:

- AI: Artificial Intelligence
- API: Application Programming Interface
- CPU: Central Processing Unit
- GPU: Graphics Processing Unit
- GUI: Graphical User Interface
- ML: Machine Learning
- RAM: Random Access Memory
- RL: Reinforcement Learning
- SQL: Structured Query Language
- UI: User Interface
- UX: User Experience

Smart Slice Selection in 5G: Unleashing the Power of Machine Learning

A Mohan Kumar
Student , Department of Computer
Science Engineering
PES UNIVERSITY
Bengaluru , Karnataka , India
amohankumar2002@gmail.com

Suhas TJ
Student , Department of Computer
Science Engineering
PES UNIVERSITY
Bengaluru , Karnataka , India
suhas30tj@gmail.com

Allu Vineela Sravanti
Student , Department of Computer
Science Engineering
PES UNIVERSITY
Bengaluru , Karnataka , India
alluvineelasravanti@gmail.com

SathyanaRaya RK
Student , Department of Computer
Science Engineering
PES UNIVERSITY
Bengaluru , Karnataka,India
sathyark652@gmail.com

Dr Vinodha K
Associate Professor , Department of
Computer Science Engineering
PES UNIVERSITY
Bengaluru , Karnataka , India
vinodhak@pes.edu

Abstract — In the rapidly advancing realm of 5G networks, the proliferation of 5G technologies has introduced a critical feature known as network slicing, allowing the creation of specialized virtual networks tailored to diverse applications. However, the challenge lies in ensuring Quality of Service(QoS) metrics for each slice in dynamic scenarios. To address this, a comprehensive analysis was conducted, exploring traditional algorithms, plain supervised models, and deep learning techniques using the SimPy simulation tool to construct a basic architecture for a traditional approach. Traditional algorithms and supervised models serve as benchmarks, highlighting their constraints in the dynamic 5G landscape. The study aims to compare traditional and machine intelligence approaches, focusing on determining which method offers the most accurate allocation of the slices. This research endeavors to provide insights into the efficiency of different approaches in addressing the intricate challenges of slice allocation in the context of 5G network slicing, contributing to the advancement of this transformative technology

Keywords — 5G, Network Slicing, virtual networks, Quality of Service(QOS), traditional algorithms, supervised models, deep learning techniques, SimPy Simulation.

I. INTRODUCTION

The landscape of communication systems underwent a seismic shift with the advent of 5G technology, introducing unparalleled advancements in speed, latency, and overall efficiency, surpassing its predecessor, 4G. Among the groundbreaking innovations within the domain of 5G, the concept of "network slicing" stands as a transformative paradigm. This pioneering approach allows for the creation of distinct and virtualized network segments, finely tailored to meet the requirements of each application by allocating the right network slice type which are enhanced mobile broadband (eMBB), ultra-reliable low latency communications (URLLC), and massive machine type communications (mMTC). Each slice assigned is meticulously designed to fulfill unique service-level agreement (SLA) requirements, effectively dedicating resources and services for optimized network efficiency.

However, the progress to 5G poses challenges in resource allocation and meeting Quality of Service (QoS) demands. Traditional algorithms and manual approaches, inherent in current network slicing methodologies, confront difficulties in adapting to the evolving and diverse requirements of 5G networks. The study at hand seeks to bridge this gap by contrasting conventional algorithms with cutting-edge machine learning approaches. Embracing machine learning techniques offers a promising avenue for adaptive and responsive slice allocation, poised to significantly enhance the efficacy of network slicing and elevate the overall QoS within the dynamic landscape of 5G networks.

Machine intelligence, which encompasses both machine learning and artificial intelligence techniques, has become a viable option for managing the heterogeneity of 5G wireless networks and addressing the slice allocation challenges associated with network slicing in order to enhance the quality of service provided to users[2]. Several studies have already demonstrated the effectiveness of using machine learning algorithms in designing efficient network slicing schemes. These studies have shown that machine learning algorithms can effectively collect and analyze data, extract optimal weight features, and classify slices based on their specific requirements. Moreover, machine intelligence enables adaptability and robustness in a dynamic networking environment.

So far, the traditional efforts in addressing this issue have been limited to either infrastructure slicing or spectrum resource slicing, with the focus on traditional mobile services[3]. On the other hand, network slicing assisted by intelligent learning offers a more comprehensive and effective approach. This approach enables the cost-effective creation of multiple logical networks within a single physical network infrastructure. These logical networks, known as network slices, can be customized to meet the unique requirements of different applications and services in a very accurate way with the help of machine intelligence. Efficiency and performance that meet Quality of Service (QoS) requirements are pivotal for a Network Slice. In our demonstration, we showcase the traditional approach slice allocation alongside the machine learning-based allocation. The emphasis lies in elucidating why advancing to a machine learning-based allocation is fundamental to our problem. We provide a side-by-side comparison of slice allocation

processes: one through the conventional approach and the other through the machine learning-based approach. Through this comparative analysis, we aim to discern the superior type of slice allocation. We not only highlight the advantages of the chosen approach in terms of QoS but also underscore how its adoption enhances overall system performance and efficiency. In doing so, our goal is to offer a clear understanding of the benefits of opting for a machine learning-based resource allocation strategy in the context of Network Slicing.

Furthermore, network slicing assisted by intelligent learning allows for dynamic slice allocation, which ensures that slices are allocated in real-time to meet the changing demands of each requirement. This has the potential to greatly enhance the quality of service provided to users by optimizing slice allocation and minimizing congestion and latency issues.

II. LITERATURE REVIEW

In recent years, the concept of network slicing has gained considerable attention due to its potential to enable operators to partition a single physical network into multiple virtual networks, each tailored to specific applications or services. R. Li et al.[1] proposed a deep reinforcement learning (DRL) method particularly Deep Q-Learning (DQL) for resource management in network slicing. Their approach, based on a multi-agent system employing deep neural networks, demonstrated adaptability to dynamic network environments, learning from experiences to optimize resource allocation, and scalability for large-scale networks.

Additionally, M. Yan.[3] highlighted vital challenges in network slicing, stressing the need for resource isolation, service differentiation, and dynamic resource management. It explores virtualization technologies such as Software-defined Networking (SDN) and Network Function Virtualization (NFV) to customize virtual networks for various applications. Introducing iRSS, an intelligent 5G RAN slicing strategy, the collaborative learning framework integrates LSTM for large-scale traffic prediction and A3C for small-scale resource scheduling. This yields notable enhancements in accumulated reward and resource utilization, showcasing iRSS's effectiveness in optimizing performance and resource allocation.

Another significant contribution came from F. Song.[4], who discussed dynamic virtual resource allocation for 5G and beyond network slicing. Their approach employed Software-Defined Networking and Network Function Virtualization to allocate and deallocate resources in real-time, catering to the changing needs of applications and services. The evaluation showcased efficient resource utilization and different service quality levels.

Furthermore, A. Huang.[5] proposed a multi-agent reinforcement learning-based approach for resource management in end-to-end network slicing. Their method optimized resource allocation, adapted to network changes, and provided varying levels of service quality, security, and reliability to different applications and services.

L. U. Khan.[6] discussed the challenges and opportunities presented by network slicing in 5G networks. They emphasized the ability of network slicing to provide customized services, optimize resource utilization, and

enable flexible service-level agreements (SLAs), thereby impacting various industries such as healthcare and transportation.

Finally, A. Thantharate.[7] discussed the security concerns in the 5G network and presented a deep learning neural network model to create a robust Network Slicing framework to combat DDoS attacks filtering the malicious UE connections to the 5G network.

Collectively, these studies have significantly contributed to advancing resource management, security, and efficiency in network slicing, paving the way for the realization of diverse and tailored network services in the 5G and beyond era.

III. OVERVIEW OF METHODS USED

The traditional 5G network slicing approach employs diverse resource allocation methods, with our discussion centered on two specific algorithms.

A. Dynamic Allocation: This real-time resource allocation method dynamically adjusts resources based on evolving application demands, ensuring adaptive and efficient utilization.

B. Priority-based allocation: This resource allocation method assigns resources based on application priority levels, prioritizing higher-priority applications for preferential access. The strategy aligns resource allocation with the criticality or importance of each application in the system.

By combining both algorithms, we integrated a hybrid approach that incorporates both dynamic resource allocation, responsive to the real-time demands of applications, and priority-based allocation, ensuring resources are assigned in accordance with application priority levels. This synergy creates a comprehensive resource allocation strategy that adapts to evolving requirements while strategically prioritizing higher-criticality applications for optimal system performance.

Input:

Set of network slices NN

Set of network switches SS

1. Sort NN based on their priority in descending order.

2. For each network slice n in sorted NN:

- Calculate required bandwidth(R_b) based on QoS requirements.
- Iterate through switches s in SS to allocate bandwidth.
- if $R_b \leq s.\text{capacity}$
- $A_b <- \min(R_b, s.\text{capacity})(\text{allocated bandwidth})$
- $s.\text{capacity} <- s.\text{capacity} - A_b$
- $n.\text{alloctaed_bandwidth} <- A_b$
- print "Allocated A_b bandwidth to n.name".
- if $A_b >= 0.8 * s.\text{capacity}$, adjust $s.\text{capacity} <- s.\text{capacity} - 0.1 * A_b$
- Break

- else,print“insufficient bandwidth to allocate”

C. Stacking Classifier:

- Ensemble learning technique that combines predictions from multiple models.
- Uses a meta-model to make the final prediction.

IV. DATASET

The dataset comprises key attributes defining parameters essential to 5G network slicing, including User Device Type, Duration, Packet Loss Rate, Packet Delay Budget, Bandwidth, Delay Rate, Speed, Jitter, Modulation Type, and Slice Type. Exploratory data analysis aimed to identify the most influential attribute for optimizing resource allocation. This optimization is vital for meeting diverse service-level agreements across network slices, thereby enhancing 5G network operations' efficiency and effectiveness.

User Device Type:	This attribute indicates the type of device that is used to connect to the network slice.
Duration (sec):	The duration of the network slice connection, in seconds.
Packet Loss Rate (Reliability):	The percentage of packets that are lost during the network slice connection.
Packet Delay Budget (Latency (ms)):	The maximum amount of latency that is allowed for the network slice connection.
Bandwidth (GHz):	Bandwidth is the amount of data that can be transferred per unit Time.
Delay Rate (Mbps):	Delay rate is the rate at which packets are delayed during the network slice connection.
Speed (Mbps):	Speed is the maximum amount of data that can be transferred per unit of time.
Jitter (ps):	Jitter is the variation in the delay of packets during the network slice connection
Modulation Type:	Modulation is the process of encoding data into a signal that can be transmitted over a network

Fig. 1. The Dataset of the Proposed Approach.

V. IMPLEMENTATION

Each network slice is crafted for specific purposes. Enhanced Mobile Broadband (eMBB) prioritizes high data rates and low latency, catering to applications like video streaming and virtual reality. Ultra-Reliable Low Latency Communication (URLLC) focuses on exceptional reliability and ultra-low latency for critical applications such as automated vehicles and emergency services. Massive Machine Type Communication (mMTC) is designed for connection density and energy efficiency, suitable for scenarios like the Internet of Things (IoT). These slices are tailored to support diverse services, coexisting seamlessly on the same infrastructure, meeting unique performance criteria for each category.

In aligning with specific user requirements, the allocation of network slices becomes imperative. As previously detailed in Section III, a hybrid methodology incorporating both dynamic and priority approaches has been adopted for resource allocation. It has been discerned that resource allocation and the selection of an appropriate network slice type are intrinsically interconnected. To explore this relationship, we conducted simulations in a foundational environment designed for resource allocation and slice type

selection. However, our observations indicate a challenge: while resource allocation effectively caters to user requirements, it encounters limitations in identifying an appropriate slice type in areas where there are requirements beyond the predefined fixed criteria. This underscores the need for further refinement in traditional approaches to achieve a more adaptive and responsive system.

Therefore,to address the above challenge we explored machine intelligence methods,

A network administrator or algorithmic technique is used to implement network slicing, which is a comprehensive approach. However, if the approach is human-based, there are numerous drawbacks, including limited speed, error-proneness, and an inability to adapt to abrupt changes in the network environment. Lack of intuition and contextual understanding, vulnerability to unforeseen scenarios, and many other problems arise when using an algorithmic method.

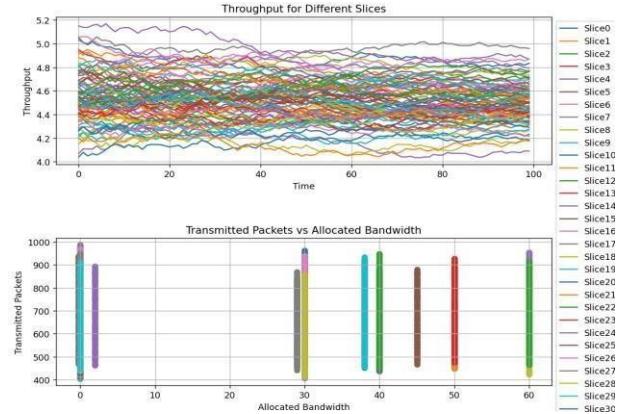


Fig. 2. Dynamic Resource Allocation

Fig.2. demonstrates the dynamic adaptation to evolving network conditions. It visualizes real-time adjustments in resource allocation to slices, catering to their individual requirements. This promotes adaptability and responsiveness in the allocation system

2) **Ensemble Techniques:** Stacking and Voting Classifier approaches were employed to harness the strengths of multiple models. The Stacking Classifier demonstrated superior performance, indicating the potential benefits of combining diverse models.

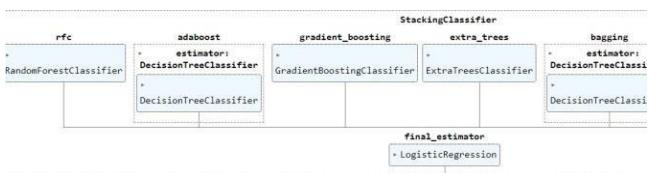


Fig. 1. Output of Stacked Classifier.

VI. RESULTS AND FINDINGS

Model Performance: Random Forest, AdaBoost, and Gradient Boosting classifiers exhibited competitive testing accuracies, highlighting their effectiveness for the given task. On the contrary, the MLP Classifier showed comparatively lower accuracy, suggesting the need for further optimization.

Stacking Classifier: The Stacking Classifier emerged as the top-performing model, showcasing the effectiveness of combining multiple models to achieve enhanced predictive accuracy.

On the other hand, the MLP Classifier exhibited lower accuracy compared to other models. This could be attributed to factors such as the need for further hyperparameter tuning or potential sensitivity to the specific characteristics of the dataset.

The performance of various machine learning models was evaluated on the dataset, and the testing accuracies were recorded.

Here are the results:

Algorithm	Accuracy
Random Forest Classifier	72.67%
AdaBoost Classifier	72.50%
Gradient Boosting Classifier	72.67%
Extra Trees Classifier	71.67%
Bagging Classifier	70.00%
Decision Tree Classifier	65.67%
MLP Classifier	41.50%
Stacking Classifier	73.00%
Hard Voting Classifier	71.17%
Soft Voting Classifier	69.33%

VII. DISCUSSIONS

By examining various resource allocation strategies, the combination of Priority-Based Allocation, and Dynamic Resource Allocation algorithms, the intrinsic dynamism of 5G architecture becomes evident. This mirrors the uncertainties found in real-world, evolving requirements. This observation propels us to simulate and test our proposed hybrid approach (dynamic resource allocation algorithm and priority based allocation) using SimPy, harnessing its capabilities to replicate the unpredictable conditions of a 5G network.

Through simulation, our goal is to assess the adaptability and performance of our algorithm in a controlled yet realistic environment. We seek insights into its effectiveness in addressing the dynamic nature of resource demands, akin to challenges encountered in real-world 5G networks. In our experimentation, we considered four distinct networks, each subjected to diverse Quality of Service (QoS) requirements. Within each network, a comprehensive test involved five strategically positioned base stations. Clients seamlessly connect to the nearest base station within their coverage area.

The success of the Stacking Classifier encourages further exploration of ensemble techniques and model combinations. Future work may involve fine-tuning hyperparameters, experimenting with additional models, and incorporating advanced deep learning architectures for improved results.

Continuous Improvement: Machine learning models should be viewed as dynamic entities, subject to refinement and adaptation. Regular updates based on new data, changing patterns, or evolving requirements are essential for maintaining the relevance and accuracy of the models over time.

VIII. FUTURE WORK

- A. Extending the study to predict traffic and enhance security gains new dimensions within the context of 5G technology. The convergence of predictive analytics with the capabilities of 5G networks provides a transformative opportunity to redefine how urban dynamics are understood and managed. In the realm of traffic prediction, the integration of 5G connectivity offers unprecedented speed and bandwidth, facilitating seamless real-time transmission of large volumes of data. This enables the constant updating and refinement of predictive models, with low latency enhancing the responsiveness of traffic prediction systems to dynamic road conditions. Researchers can explore traffic patterns with unparalleled precision, accommodating the nuances of urban mobility.

B. Simultaneously, the intersection of 5G technology with security prediction introduces innovative possibilities. Increased connectivity and data transfer speeds enable the integration of diverse security sensors and surveillance devices, creating a comprehensive network for monitoring and analysis. Machine learning models can leverage 5G's high-speed data streams to detect anomalies, identify potential threats, and enhance situational awareness in urban security systems. The reliability and stability of 5G contribute to the resilience of these systems, ensuring continuous and uninterrupted surveillance. Considering the interplay between traffic and security in a 5G-enabled environment creates an opportunity for synergy, fostering a holistic understanding of urban dynamics and ensuring an adaptive and responsive urban ecosystem. However, challenges related to 5G implementation, such as data security and privacy, must be addressed, emphasizing collaboration with stakeholders and staying attuned to the evolving landscape of 5G technology.

IX. REFERENCES

- [1] R. Li et al., "Deep Reinforcement Learning for Resource Management in Network Slicing," in IEEE Access, vol. 6, pp. 74429-74441, 2018, doi: 10.1109/ACCESS.2018.2881964.
- [2] Y. Kim and H. Lim, "Multi-Agent Reinforcement Learning-Based Resource Management for End-to-End Network Slicing," in IEEE Access, vol. 9, pp. 56178-56190, 2021, doi: 10.1109/ACCESS.2021.3072435. (RP-5)
- [3] M. Yan, G. Feng, J. Zhou, Y. Sun and Y. -C. Liang, "Intelligent Resource Scheduling for 5G Radio Access Network Slicing," in IEEE Transactions on Vehicular Technology, vol. 68, no. 8, pp. 7691-7703, Aug. 2019, doi: 10.1109/TVT.2019.2922668.
- [4] F. Song, J. Li, C. Ma, Y. Zhang, L. Shi and D. N. K. Jayakody, "Dynamic Virtual Resource Allocation for 5G and Beyond Network Slicing," in IEEE Open Journal of Vehicular Technology, vol. 1, pp. 215-226, 2020, doi: 10.1109/OJVT.2020.2990072.
- [5] A. Huang, Y. Li, Y. Xiao, X. Ge, S. Sun and H. -C. Chao, "Distributed Resource Allocation for Network Slicing of Bandwidth and Computational Resource," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9149296.
- [6] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han and C. S. Hong, "Network Slicing: Recent Advances, Taxonomy, Requirements, and Open Research Challenges," in IEEE Access, vol. 8, pp. 36009-36028, 2020, doi: 10.1109/ACCESS.2020.2975072.
- [7] A. Thantharate, R. Paropkari, V. Walunj, C. Beard and P. Kankariya, "Secure5G: A Deep Learning Framework Towards a Secure Network Slicing in 5G and Beyond," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0852-0857, doi: 10.1109/CCWC47524.2020.9031158.
- [8] S. Rathore, J. H. Park and H. Chang, "Deep Learning and Blockchain-Empowered Security Framework for Intelligent 5G-Enabled IoT," in IEEE Access, vol. 9, pp. 90075-90083, 2021, doi: 10.1109/ACCESS.2021.3077069.
- [9] V. P. Kafle, P. Martinez-Julia and T. Miyazawa, "Automation of 5G Network Slice Control Functions with Machine Learning," in IEEE Communications Standards Magazine, vol. 3, no. 3, pp. 54-62, September 2019, doi: 10.1109/MCOMSTD.001.1900010.
- [10] X. Li et al., "Network Slicing for 5G: Challenges and Opportunities," in IEEE Internet Computing, vol. 21, no. 5, pp. 20-27, 2017, doi: 10.1109/MIC.2017.3481355.



PES
UNIVERSITY
CELEBRATING 30 YEARS

CAPSTONE PROJECT 2024

Title: Smart Slice Selection in 5G: Unleashing the Power of ML

Domain: NETWORKING AND MACHINE LEARNING

Batch No: 100



Dr. Vinodka K
Supervisor



A Mohan Kumar
PES2UG20CS002



Allu Vineela Sravanti
PES2UG20CS036



Sathyanarayana RK
PES2UG20CS315

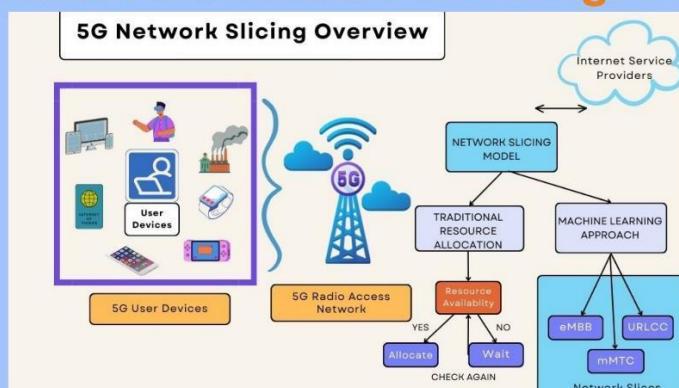


Suhas TJ
PES2UG20CS355

Abstract:

The study aims to compare Traditional and Machine Intelligence approaches, focusing on determining which method offers the most accurate resource allocation to the slices. This research endeavors to provide insights into the efficiency of different approaches in addressing the intricate challenges of resource allocation in the context of 5G network slicing, contributing to the advancement of this transformative technology.

Architectural Flow / Data Flow Diagram:



Results and Discussion:

- Used Dynamic Resource Allocation for testing in 5G Architecture. The results of 4 Networks in simulated environment exhibit variations in their base station structures and configurations.
- Applied various ML classifiers and AdaBoost performed best with (73%), followed closely by Random Forest (71.67%), Later on Selected models were stacked to adjust in various scenarios.