Addis Ababa Institute of Technology
College of Technology and Built Environment
School of Electrical and Computer Engineering
Telecommunication Network Engineering Graduate Program

Mapping QoS to QoE Using Hidden Markov Models: A Case Study on Virtual Internet Service Provider Networks

by:
Betelehem Getu

Advisor:
Dr. -Ing. Dereje Hailemariam

A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of the Requirement for the Degree of Master's Science in Telecommunication Network Engineering

June, 2025

Addis Ababa University
Addis Ababa Institute of Technology
School of Electrical and Computer Engineering
Telecommunication Network Engineering Graduate Program

Mapping QoS to QoE Using Hidden Markov Models: A Case Study on Virtual Internet Service Provider Networks

by:

Betelehem Getu

Approval by Board of Examiners	Signature	Date
Chairman, School Graduate Comittee:	_____	_____
Advisor's Name:	_____	_____
Internal Examiner's Name:	_____	_____
External Examiner's Name:	_____	_____

Declaration

I hereby declare that this thesis is my own original work and has not been submitted to any other institution for the award of a degree or diploma. To the best of my knowledge, it does not contain any material previously published or written by another person without proper acknowledgment, and all sources and materials used are duly cited and acknowledged.

Name of the Student	Signature	Date
_____	_____	_____

This thesis has been submitted for examination with my approval as the university advisor.

Name of the Advisor	Signature	Date
_____	_____	_____

Addia Ababa, Ethiopia

ABSTRACT

For Internet Service Providers (ISPs) and virtual ISPs (vISPs), accurately understanding and measuring Quality of Experience (QoE) is essential, yet doing so is a challenging task because they have relied solely on technical Quality of Service (QoS), which fails to capture the true QoE perceived by users due to the user-centric, individual, multidimensional, and multisensorial nature of QoE. Therefore, this has led to the evolution of existing predictive models that map QoS to QoE. However, they are often limited in capturing sequential patterns or hidden transitions in user-perceived quality over time.

This thesis proposes a Hidden Markov Model (HMM)-based approach to model the mapping between QoS parameters and QoE metrics using objective data and a composite of the two (objective and subjective data) for both ethio telecom and vISPs (Websprix IT Solution PLC and ZERGAW Cloud). The Baum-Welch algorithm was used to train the model, and its performance is compared against Support Vector Machines (SVM) and Random Forest (RF) models.

The results of the study convincingly demonstrate the performance scores of the HMM prediction model with 99% RF with 96%, and SVM with 71% accuracy, which outperforms the SVM and RF models. Also, HMM had around 100% in all precision, recall, and F1-score, particularly in scenarios with high network variability characteristic of vISP networks. These findings highlight the potential of HMMs for improving QoE prediction in service provider networks and supporting more user-centered service optimization strategies.

Key Words: QoS, QoE, Hidden Markov Model, Support Vector Machine, Random Forest, ISP, vISPs

Acknowledgments

First and foremost, I express my deepest gratitude to the Almighty God. The completion of this research would not have been possible without His strength and guidance.

I would also like to express my heartfelt thanks to my advisor, Dr. -Ing. Dereje Hailemariam, for his valuable guidance, insightful feedback, and consistent support throughout the course of this study. His mentorship has been instrumental in shaping both the direction and quality of this work.

I am deeply grateful to my parents, siblings, and friends for their unconditional love, encouragement, prayer, and constant support. Their belief in me has been a continuous source of motivation.

My appreciation also goes to Abayneh Mekonen, Amel Salem, and Abera Dibaba for their support; my thanks extend to all instructors and colleagues who provided support and encouragement along the way.

Table of Contents

Dedication iv

Acknowledgements iv

Table of Contents v

List of Tables vii

List of Figures viii

List of Abbreviations x

Chapter 1: **Introduction** 1
- 1.1 Background . . . 1
- 1.2 Literature Review . . . 4
- 1.3 Statement of the Problem . . . 7
- 1.4 Objective . . . 8
  - 1.4.1 General Objective . . . 8
  - 1.4.2 Specific Objectives . . . 8
- 1.5 Scope and Limitation . . . 9
- 1.6 Methodology . . . 9
- 1.7 Significance of the Research . . . 11
- 1.8 Thesis Organization . . . 11

Chapter 2: Overview of Virtual Internet Service Providers 13
- 2.1 Definition of vISPs . . . 13
  - 2.1.1 Technical Model for vISPs . . . 15
  - 2.1.2 Key Roles and Benefits of vISPs . . . 17
  - 2.1.3 Challenges of vISPs . . . 19
- 2.2 QoS and QoE in Influencing Factors in vISPs . . . 20
  - 2.2.1 QoS Factors . . . 20
  - 2.2.2 QoE Factors . . . 22
  - 2.2.3 QoE Modeling Techniques . . . 23

Chapter 3: Machine Learning for QoS to QoE Mapping 27
- 3.1 Hidden Markov Model . . . 27
  - 3.1.1 Markov Chain . . . 27

3.1.2 Hidden Markov Models . . . 31
3.1.3 Components of HMM parameter . . . 32
3.1.4 Fundamental Problems in HMM . . . 36
3.1.5 Common Solutions to HMM Problems . . . 37
3.1.6 Applications of HMMs . . . 41
3.2 Support Vector Machines . . . 43
3.2.1 Types of SVMs . . . 44
3.2.2 Kernel Functions . . . 46
3.2.3 SVM Hyperparameters . . . 47
3.3 Random Forest Algorithm . . . 49
3.3.1 Random Forest Learning Techniques . . . 50
3.3.2 Random Forest Hyperparameters . . . 53

Chapter 4: Methods for QoS to QoE Mapping 54
4.1 vISPs System Model . . . 54
4.2 Data Collection . . . 56
4.3 Data Preprocessing . . . 57
4.3.1 Feature Selection . . . 58
4.3.2 Hidden State Formation . . . 60
4.3.3 Observation Sequence Formation . . . 63

Chapter 5: Results and Discussion 66
5.1 Model Training and Validation . . . 66
5.1.1 Transition Probability Matrix . . . 67
5.1.2 Emission Probability Matrix . . . 69
5.1.3 The Initial Probability Distribution . . . 69
5.1.4 Hyperparameter Optimization . . . 71
5.2 Model Evaluation . . . 73
5.2.1 Accuracy . . . 73
5.2.2 Precision . . . 74
5.2.3 Recall . . . 74
5.2.4 F1 Score . . . 75
5.3 Hidden State Prediction . . . 75
5.4 Comparison of Predicted Results . . . 78

Chapter 6: Conclusions and Future Work 85
6.1 Conclusions . . . 85
6.2 Future Work . . . 86

Appendix A: Comparision Results 87

Bibliography 89

List of Tables

4.1 Service provider and sample size 57
4.2 QoS and QoE as hidden states based on HMM concept 61
4.3 Hidden states of the HMM 61

5.1 Grid search results for different service providers 72

List of Figures

1.1 Methodology for predicting QoE . . . 10

2.1 ISP Vs vISPs . . . 14
2.2 ethio telecom and vISPs . . . 14
2.3 vISP technical model [15] . . . 16
2.4 vISP technical model 2 [15] . . . 17
2.5 Key roles and benefits of vISPs . . . 19
2.6 QoE influence factors . . . 23
2.7 Technical data to MOS value mapping technique . . . 25
2.8 Composite of both data to MOS mapping technique . . . 25
2.9 Technical to subjective data mapping technique . . . 26

3.1 Discrete-time Markov Chain . . . 29
3.2 Hidden Markov Model . . . 34
3.3 Key HMM algorithms . . . 41
3.4 SVM hyperplane [41] . . . 44
3.5 SVM example of linearly separable data [31] . . . 45
3.6 SVM example of nonlinearly separable data with the kernel trick [31] . . . 46
3.7 C hyperparameter [41] . . . 48
3.8 Gamma hyperparameter [41] . . . 49
3.9 Random Forest Algorithm [40] . . . 50
3.10 Bootstrap aggregating technique [42] . . . 51
3.11 RF boosting technique [42] . . . 52

4.1 HMM system model for predicting QoE . . . 55
4.2 SVM and RF system model for predicting QoE . . . 56
4.3 Feature selection for ethio telecom . . . 59
4.4 Feature selection for Websprix . . . 59
4.5 Feature selection for Zergaw . . . 60
4.6 Transition probabilities . . . 63
4.7 Optimal number of clusters using elbow method for ET . . . 63
4.8 Optimal number of clusters using elbow method for Websprix . . . 64
4.9 Optimal number of clusters using elbow method for Zergaw . . . 64
4.10 Formation of a unique number of observation symbols . . . 65

5.1 Baum Welch's model convergence . . . 67
5.2 Transition probability matrix for ethio telecom, Websprix and Zergaw . . . 68
5.3 Emission probability matrix for ethio telecom, Websprix and Zergaw . . . 69

5.4 Initial probability distribution for ethio telecom, Websprix and Zergaw . . 70
5.5 K - fold cross validation model training [38] 71
5.6 QoE prediction output and model performance of HMM 77
5.7 Confusion matrix for ethio telecom, Websprix, and Zergaw 79
5.8 Confusion matrix for ethio telecom, Websprix, and Zergaw 80
5.9 Confusion matrix for ethio telecom and Websprix 80
5.10 Confusion matrix for Zergaw 81
5.11 Model prediction evaluation for ethio telecom and Websprix 81
5.12 Model prediction evaluation for Zergaw 82
5.13 Model prediction evaluation for ethio telecom and Websprix 82
5.14 Model prediction evaluation for Zergaw 83
5.15 Model prediction evaluation for ethio telecom and Websprix 83
5.16 Model prediction evaluation for Zergaw 84

A.1 Transition probability, emission probability, and initial distribution probability of ethio telecom 87
A.2 Transition probability, emission probability, and initial distribution probability of Websprix 87
A.3 Transition probability, emission probability, and initial distribution probability of Zergaw 87
A.5 k fold cross validation 88
A.6 Decision boundery for ethio telecom, Websprix and Zergaw 88

List of Abbreviations

5G	Fifth Generation
ACR	Absolute Category Rating
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
BW	Bandwidth
bps	Bits per second
CAPEX	Capital Expenditure
CES	Customer Effort Score
CR	Cognitive Radio
CSAT	Customer Satisfaction
DCR	Degradation Category Rating
DL	Download
DNA	Deoxyribonucleic Acid
EM	Expectation Maximization
E-step	Expectation Step
GA	Genetic Algorithm
Gbps	Gigabits per second
GE	Gigabit Ethernet
HMM	Hidden Markov Model
IP	Internet Protocol
IPTV	Internet Protocol Television
ISP	Internet Service Provider
ITU	Ineternational Telecommunication Union
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
Kbps	Kilobits per second
KNN	K-Nearest Neighbor
KPI	Key Performance Indicator
LTE	Long Term Evolution
M5P	M5 Model Tree with Pruning
MAE	Mean Absolute Error
Mbps	Megabits per second
MC	Markov Chain
MCD	Ministry of Communication and Digital Technologies
MDU	Multi-Dwelling Unit

MOS	Mean Opinion Score
MP	Markov Property
MPLS	Multi-Protocol Label Switching
M-step	Maximization Step
MSAG	Multi-Service Access Gateway
MSAN	Multi-Service Access Node
MSE	Mean Square Error
NETEM	Network Emulator
NLP	Natural Language Processing
NN	Neural Networks
PCA	Principal Component Analysis
PRTG	Paessler Router Traffic Grapher
QoE	Quality of Experience
QoS	Quality of Service
RMSE	Root Mean Squared Error
RF	Random Forest
SDN	Software Defined Networking
SLAM	Simultaneous Localization And Mapping
SSE	Sum of Square due to Error
SSIM	Structural Similarity Index Measure
SVM	Support Vector Machine
UL	Upload
VAS	Value Added Services
vISP	Virtual Internet Service Provider
VOD	Video On Demand
VPN	Virtual Private Network
VQM	Video Quality Metric
VoIP	Voice over IP
WEKA	Waikato Environment for Knowledge Analysis
xDSL	Digital Subscriber Line (various types)
xPON	Passive Optical Network (various types)

Chapter 1: **Introduction**

The growing reliance on Internet-based services has made user satisfaction one of the key performance indicators for Internet Service Providers (ISPs) and virtual Internet Service Providers (vISPs) that operate virtually without their own infrastructure. While traditional metrics such as latency and packet loss provide a notion about network performance, they often fail to capture the user's actual experience. This gap between objective Quality of Service (QoS) and subjective Quality of Experience (QoE) is a challenge as well as an opportunity for intelligent modeling. This study investigates the use of Hidden Markov Models (HMM) for predicting QoE from QoS data within the setting of a Virtual ISP (vISP) environment. The chapter sets the stage by outlining the background, problem statement, research objectives, and methodological approach, while determining the relevance and significance of the study in advancing user-centric network evaluation.

1.1 Background

Until recently, QoS in telecommunication systems was mainly evaluated from a technical perspective, focusing on several measurable factors such as throughput, available bandwidth, delay, error probability, jitter, and packet loss [1]. In 1994, according to the International

Telecommunication Union (ITU) recommendation ITU-T Rec. E.800, quality of service was defined as "Collective effect of service performance which determines the degree of satisfaction of a user of the service." Markaki redefined it [2] as the, "Capability of a network to provide better service to selected network traffic ... described by the following parameters: delay and jitter, loss probability, reliability, throughput and delivery time." As network usage grew more diverse and user expectations evolved, researchers and standards bodies began to consider user perception as a complementary dimension—leading to the concept of QoE.

In contrast, [3] presents a commonly used definition of QoE as "Overall acceptability of an application or service as perceived subjectively by the end-user ... includes the complete end-to-end system effects ... maybe influenced by user expectations and context."

In general, QoS refers to how well a network meets service-level requirements, while QoE reflects how well the service performance aligns with user expectations. QoS is a technical measure, typically expressed in terms of packet loss, jitter, and bandwidth. QoE, on the other hand, is subjective and difficult to quantify, often assessed using a five-level Mean Opinion Score (MOS) scale, where 5 represents "excellent" and 1 denotes "bad" [4].

Although QoS and QoE are closely linked, they are not the same. QoE is often regarded as a non-parametric and nonlinear function of QoS, meaning that optimizing network parameters does not always lead to improved user satisfaction. This is due to the fact that network-level measurements and the random nature of human behavior do not directly capture user-perceived quality. While good QoS is often a prerequisite for good QoE, the relationship between the two is complex and not always predictable. Understanding this relationship is essential for network operators seeking to improve user

experience.

Ensuring a consistently high QoE presents a significant challenge due to the dynamic and unpredictable nature of network environments. As a result, there is growing interest in developing predictive models that can effectively map QoS parameters to QoE, enabling proactive network resource management and improved user satisfaction. QoS-to-QoE mapping is the process of translating measurable QoS indicators into meaningful QoE metrics. This process is essential for understanding how network performance affects user experience and for designing strategies that enhance QoE.

To address these challenges, statistical models capable of handling temporal and hidden dynamics are increasingly applied in QoS–QoE mapping tasks. This thesis investigates the prediction of QoE by leveraging QoS-to-QoE mapping using Hidden Markov Models (HMMs). HMMs are statistical models designed to analyze observed sequences that evolve over discrete time steps. These sequences may be countably or continuously distributed [5].

In the context of predicting QoE from network metrics, HMMs offer a powerful tool for capturing hidden patterns and temporal dependencies in observed data. They model the system as a probabilistic process with unobserved (hidden) states, making them suitable for representing the uncertain and dynamic aspects of user perception and network behavior [5]. In addition to HMMs, this study considers Support Vector Machines (SVMs) and Random Forests (RFs)—two widely used machine learning algorithms.

Support Vector Machines (SVMs): SVMs are kernel-based learning methods that aim to find an optimal hyperplane for classification tasks. This is achieved by solving a convex quadratic optimization problem, which guarantees a globally optimal solution.

SVMs are non-parametric supervised algorithms, in the sense that they are not sensitive to the distribution of the underlying data. They are known to perform nicely with high-dimensional data and also with small training sets comfortably. However, their performance heavily depends on the choice of an appropriate kernel function, and they can be prone to overfitting.

Random Forests (RFs): RF is an ensemble learning method that combines multiple decision trees to improve classification accuracy. As an ensemble technique, RF reduces variance and enhances the robustness of predictions, especially when dealing with unstable classifiers. It is recognized for being relatively easy to implement—requiring minimal hyperparameter tuning—and for its ability to learn both simple and complex patterns. RFs are also more resistant to overfitting compared to boosting methods.

1.2 Literature Review

Extensive research has been conducted on modelling QoE over the past few decades. Different approaches have been developed, including subjective models based on user feedback, objective models based on mathematical relationships, and hybrid models combining subjective and objective approaches.

The study in [10] is motivated by the need for real-time prediction of user-centric QoE as metrics fail to reflect the actual user experience. Their objective was to use different machine learning algorithms to predict the QoE from the QoS and evaluate the model performance. The methodology they used involved collecting data through a mobile app that recorded user ratings along with QoS parameters, and then preprocessing the data to

extract features that fit Random Forest, Support Vector Machines, Gradient Boosting, and Neural Networks models. Then evaluated the models using metrics like Mean Absolute Error (MAE) and classification accuracy. The results showed that Gradient Boosting and RF demonstrated better performance and greater accuracy. However, the limitation of the study was the geographically limited and relatively small dataset.

The study in [11] is motivated by highly reliable fixed broadband internet services and a lack of comprehensive QoE models using different machine learning algorithms. The study is to model the QoE of fixed broadband service by using two machine learning methods (SVM and RF) by integrating QoS parameters. The methodology they used involved collecting from ethio telecom and vISPs, followed by data preprocessing involving normalization, feature selection to pinpoint key factors, hyperparameter tuning via grid search, model training and validation with cross-validation techniques, and performance assessment across subjective, objective, and hybrid QoE metrics. The results showed that with an accuracy of 92%, SVM outperforms RF for vISPs, whereas with 88% accuracy, RF outperforms for ethio telecom. Also, combining subjective and QoS metrics improves model performance, enabling effective prediction of user satisfaction based on network parameters and user data. However, the limitation of the study was the relatively small size and divergence of the dataset. Additionally, the ranges of the hyperparameters are limited, hindering further optimization.

The motivation for the study in [12] is the high mobile data demand in Addis Ababa, Ethiopia, and inefficient, real-time, and not cost-effective measurements of user satisfaction by subjective methods, which leads to the need for an objective QoE measurement model for LTE web browsing services. Their objective was to develop a neural network-based

model in MATLAB to predict QoE by identifying QoS parameters and evaluating the model. However, the limitation of the study was that it was based only on specific QoS metrics, and due to the fact that the study focused on LTE web browsing in Addis Ababa, it didn't consider all the influential factors, such as the network variation, user demographics, and context. The methodology they used includes collecting network data, followed by the identification of the important QoS parameters. Prediction of user experience (MOS) is done after training an Artificial Neural Network (ANN) model, then model performance is evaluated by using metrics like Mean Square Error (MSE) and correlation coefficient. The results showed a correlation between estimated and actual QoE of 97% model accuracy and a low MSE of 0.002, which indicates effective prediction capabilities.

Video streaming in software-defined networking (SDN) environments has a challenge of real-time QoE assessment, and the study [13] addresses this challenge to overcome the limitations of traditional subjective methods such as degradation category rating and mean opinion score, which are costly and time-consuming. The author's objective was to develop an objective QoE prediction model using different machine learning methods. They trained Decision Tree (M5P), Neural Network, K-Nearest Neighbors (KNN), and RF algorithms using full-reference video metrics, including Structural Similarity Index Measure and Video Quality Metric. Evaluate the model performance using Pearson correlation, RMSE, and cross-validation in WEKA. The results showed RF outperformed other models, while M5P also performed well. On the other hand, KNN and Neural Networks were less effective. However, the limitation of the study was its restriction on subjective testing for training data, and it is a dataset-specific evaluation that affects the model to accurately predict.

1.3 Statement of the Problem

In the telecommunications sector, ISPs continue to rely heavily on QoS metrics to monitor and manage network performance. However, these metrics do not always align with the actual experiences of end users, leading to a persistent gap between operational evaluations and perceived service quality. While this disconnect has been widely acknowledged, and several efforts have been made to map QoS to QoE using various machine learning techniques, challenges remain.

Many existing models are developed around specific services, such as video streaming or VoIP, and often overlook the broader service environment in which ISPs operate. Additionally, the temporal dynamics of user experience, an essential aspect in real-world network usage, are frequently underrepresented. Although techniques like SVMs and RFs have shown promise, they are often limited in capturing sequential patterns or hidden transitions in user-perceived quality over time.

This thesis addresses these limitations by focusing on the application of Hidden Markov Models for QoE prediction within a virtual ISP network context. Despite the potential of HMMs to model time-dependent behaviors and latent user experience states, their adoption in general ISP service environments remains limited. There is a critical need for predictive frameworks that not only infer QoE from QoS effectively but also accommodate the dynamic, multi-service nature of ISP networks. Bridging this gap will enable ISPs to move from reactive quality monitoring to proactive, user-centric service management.

1.4 Objective

1.4.1 General Objective

The general objective of this thesis is to develop and evaluate predictive models that map QoS parameters to QoE metrics using HMM, SVM, and RF within an ISP network context.

1.4.2 Specific Objectives

- To conduct a comprehensive literature review on ISP networks, QoS–QoE relationships, and the application of machine learning models for QoE prediction.
- To formulate a modeling framework that maps QoS parameters to QoE metrics using HMM, SVM, and RF algorithms.
- To collect and preprocess ISP network data suitable for QoS-to-QoE modeling.
- To identify relevant QoS and QoE features that influence model building and apply dimensionality reduction techniques, such as Principal Component Analysis (PCA) and k-means clustering, to enhance model efficiency and interpretability.
- To implement and train prediction models using HMM, SVM, and RF techniques.
- To evaluate the predictive performance of HMM, SVM, and RF models using standard validation metrics, compare their effectiveness in mapping QoS to QoE, and draw conclusions on their relative strengths and suitability within the ISP context.

1.5 Scope and Limitation

This thesis focuses on developing a predictive framework for estimating QoE by mapping QoS parameters to user experience within the context of a virtual ISP network. The study applies HMMs to capture the temporal characteristics of network performance data and infer latent states that represent user perception. To benchmark performance, the proposed framework is evaluated against two additional machine learning models: SVMs and RFs, using the same dataset and evaluation metrics.

The limitations of this thesis come from three aspects. First, the thesis is based on data collected from ethio telecom and two private vISPs and the volume of data per vISP is limited. This is partly because of the limited number of subscribers at the private vISPs. This limitation may affect the generalizability of the findings across more diverse vISPs. Second, due to time constraints and a focus on the modeling aspect, the thesis considered only three machine learning algorithms: namely, HMM, SVM, and RF. Third, the identified QoS parameters or features are limited to the network layer, while QoE-influencing factors such as user device type, content, and contextual variables are not considered.

1.6 Methodology

After the problem has been identified and the objective has been determined, the following methodology is used to address the identified problem. The methodology is presented using a visual workflow diagram shown in, Figure 1.1. The research relies heavily on observational data. First, data is collected from the ISP and vISPs. The collected

data is passed through a preprocessed process that involves data cleaning, normalization, and feature selection to ensure the data fits with the models. For HMM, hidden states and observable states are defined, and data splitting is done. Before the training of data clustering is done on the data, as needed. As for SVM and RF Grid search technique is used to choose the optimal kernel type, C, and Gamma parameter. We used the Baum-Welch algorithm for model training to estimate the HMM parameters, including initial, transition, and emission probabilities, and then the Viterbi algorithm to find the most probable sequence of hidden states, whereas SVM and RF models were trained using the selected features and hyperparameters. Once the models were trained and parameters were obtained, we predicted the user experience and the sequence of the QoE over time. Finally, model performance evaluation was conducted.

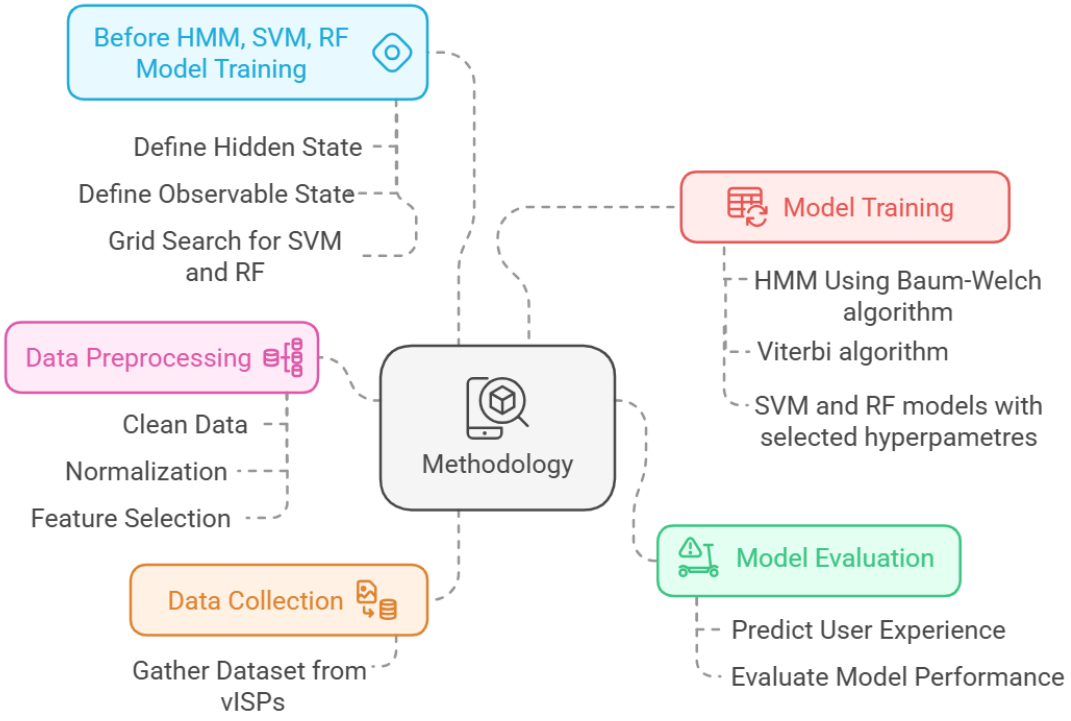

Figure 1.1: Methodology for predicting QoE

1.7 Significance of the Research

This thesis develops a novel approach based on Hidden Markov Models (HMMs) for accurate modelling and prediction of user QoE over time in a virtual ISP context. Differing from other models, this approach captures the temporal dependencies and hidden user experience states. Thus, expectations of user satisfaction from the network QoS have contributed to the framework. By correlating objective QoS indicators with subjective user perceptions, the framework achieves a more accurate and user-centric assessment of service quality, enabling ISPs to move towards the proactive rather than reactive management of service. The thesis provides a comprehensive comparison against other existing supervised machine learning algorithms (like SVM and RF), proving that temporal probabilistic modeling outperforms the capture of user experience dynamics. This information can help the service provider to better manage the network resources and hence improve the user experience. Through this study, the QoE prediction technique is extended to complicated multi-service networks by highlighting the importance of temporal and contextual factors.

1.8 Thesis Organization

The remaining parts of the thesis document are organized as follows: Chapter 2 gives an overview of vISPs, key roles, and challenges of vISPs. Chapter 3 explores how machine learning can be used to model the QoE, focusing on the use of HMM, including background

knowledge, components of HMM, assumptions used in HMM, and the three basic problems of HMM, with their solution will be discussed briefly. In addition, other machine learning algorithms SVM and RF are discussed. Chapter 4 delves into the methodology employed for designing the QoE model using HMM, SVM, and RF, giving a clear overview of each step and technique. Chapter 5 presents the results of our research and explains what those results mean and including model evaluation. Chapter 6 concludes the thesis by summarizing our key findings and considerations for future work.

Chapter 2: Overview of Virtual Internet Service Providers

This chapter provides an overview of vISPs by defining vISPs and outlining their technical models, followed by a discussion of their key roles, benefits, and challenges. Then, examines how QoS and QoE function as critical determinants of service effectiveness within vISP environments. Special focus is placed on the factors that influence user satisfaction and the performance of vISPs, with a review of established QoE modeling techniques.

2.1 Definition of vISPs

According to the definition given in the ethio telecom commercial circular number ET/MCD/P&S/71/2017 (2017), Virtual Internet Service Providers is a company that offers Internet services under its own company or brand name, while actually using the equipment and facilities of another Internet Service Provider to provide those services. From this definition, ethio telecom will sell internet service to vISPs at a wholesale level, and they will retail it for final users with or without additional value. Unlike traditional ISPs, vISPs do not own the physical network infrastructure (such as fiber optic cables or towers) but lease bandwidth or wholesale internet capacity from infrastructure providers like ethio telecom.

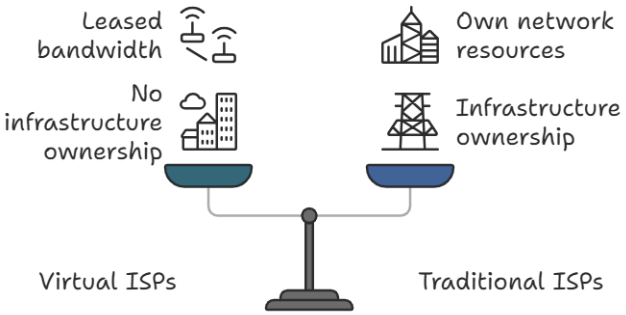

Figure 2.1: ISP Vs vISPs

As per the engagement guide on the above mentioned circular, to be a vISP, a company needs to have a Value Added Service (VAS) license and a wholesale agreement with ethio telecom and then the vISPs will compete with the wholesale service provider (ethio telecom) in the selected market segment by providing a competitive offer, provisioning, maintenance and customers services. In addition to the basic internet service, vISPs can provide additional VAS like Video on Demand, IPTV, and others, and to give these services, vISPs are expected to make some investment. The first vISP partner, WebSprix IT Solutions PLC, started operation as of April 20, 2018, and Zergaw Cloud began on September 18, 2020 [15].

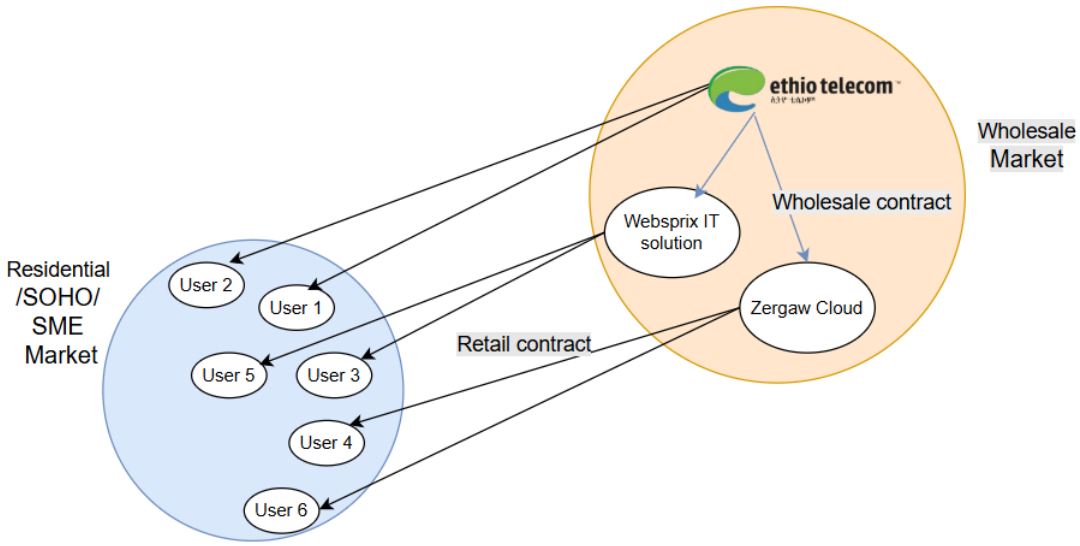

Figure 2.2: ethio telecom and vISPs

2.1.1 Technical Model for vISPs

According to the commercial circular number ET/MCD/P&S/71/2017, two service provisioning methods are proposed for partners to enter into partnerships according to their preference.

1. **vISP Technical Model:** In Figure 2.3 shown below, the wholesale internet capacity that the vISP subscribed and will retail to its end users will be connected to its service platform, which will be located in the premises of ethio telecom and connected to the ethio network. ethio telecom will provide connection up to the vISP's Access Device (MDUs, MSAGs, or modems) via fiber links, using fiber cores sourced from different ethio telecom equipment and connected to the ethio IP network. These access devices must support a wide range of technologies, including xDSL, xPON, and multiple GE cards, to meet the diverse and growing demands of customers. Beyond the access point, the vISP will be responsible for deploying the secondary network, managing internal cabling, and operating its own service platform and infrastructure to deliver end-user services [15].

   From the below technical model, ethio telecom's role:

   - Sell internet at a wholesale level, which will be a resale by ethio telecom of the bandwidth it purchases from international connection providers.
   - Providing a connection to the vISP service platform, which connects to ethio telecom's IP network.
   - Providing a connection up to vISPs access device via fiber, which connects to

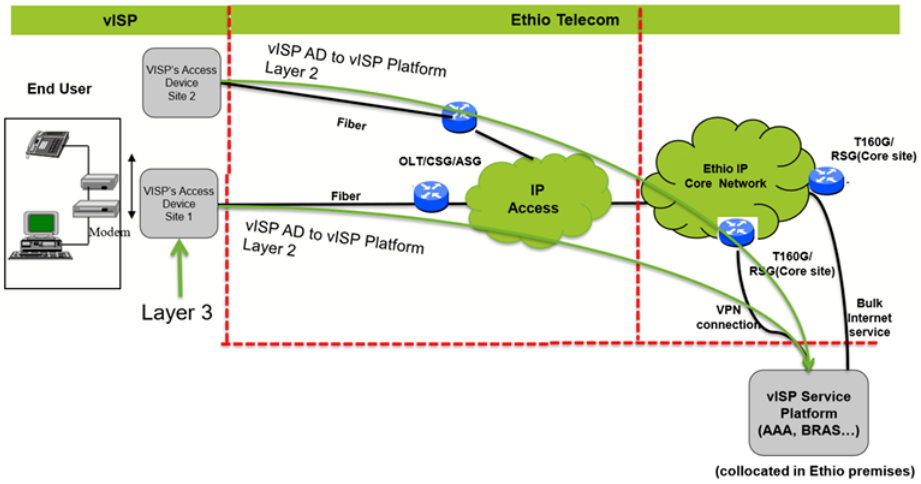

Figure 2.3: vISP technical model [15]

ethio telecom MPLS network from different equipment.

- Providing a connection to vISPs office to remotely access its service platform from office with no need for ethio data center entrance.
- Providing backup connections.
- Service maintenance when these connections are down.
- Collocation for their server, IP address per the vISPs need, and call center hosting.

2. **vISPs Technical Model 2:** In Figure 2.4 shows the technical model when vISPs use ethio telecom's secondary network infrastructure in areas where secondary network deployment is already completed, and to use the infrastructure, vISPs will pay monthly rent per line. Also, ethio telecom will be responsible for the maintenance of these secondary lines during failure [15].

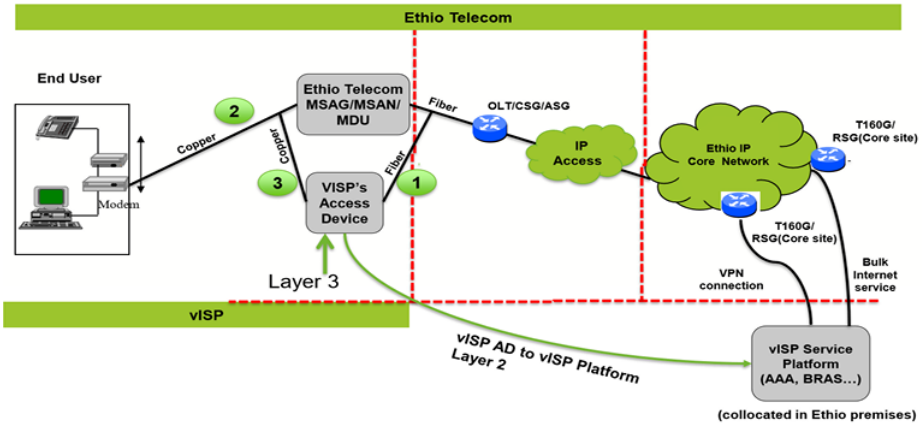

Figure 2.4: vISP technical model 2 [15]

Number 1: indicates the fiber line that is tapped from the back of ethio's access device (MSAG/MSAN/MDU) and connected to the vISP's access device.

Number 2: Indicates the secondary network deployed by ethio telecom from its access device to its end-user premises, and according to the demand.

Number 3: Indicates the secondary lines of ethio telecom that a vISP can use. From the secondary lines deployed by ethio telecom, the vISP can rent some of them and provide Internet service to its end users. These lines will not enter ethio telecom's access device; instead, they will be connected to the vISP's access device.

2.1.2 Key Roles and Benefits of vISPs

vISPs Key Roles:

- Promote its service with its own brand name and find new internet subscribers to meet its target given by ethio telecom.
- Providing Internet connectivity to residential, enterprise, and small business users to end users after their access device.

- Maintaining service failures after its access device. End users will not contact ethio telecom for provisioning and maintenance requests.
- vISPs can provide content services and may add QoS parameters to have a common quality standard to guarantee.
- The vISP will invest in a service platform, sales, delivery, maintenance, billing, and monitoring system.

vISPs Benefits

One of the important advantages of vISPs are their ability to offer their own contention ratios (the number of users sharing the same bandwidth) for different customer segments, allowing greater control over service quality and services to meet specific customer needs, offering personalized support and packages. In addition to their low costs and wide coverage, vISPs are also competitively priced. Since they do not have the expense of owning and maintaining infrastructure, they lease infrastructure, and this will allow them to significantly reduce capital expenditure and limitations to a specific network. This has been a major attraction for price-conscious consumers, who want low-cost internet without compromising on quality. Also, their cost structure allows them to offer competitive pricing and tailor their services to meet the unique needs of their customers. By renting out network capacity from ISP, vISPs can offer internet services to clients in rural or underserved locations where ISP might not be present. This has been especially useful for rural communities and developing countries, whose internet access has been a challenge. With vISPs, they are now able to enjoy being online and stay connected to the digital world. vISPs, in general, foster competition, which leads to improved service quality,

reduced pricing, and increased innovation.

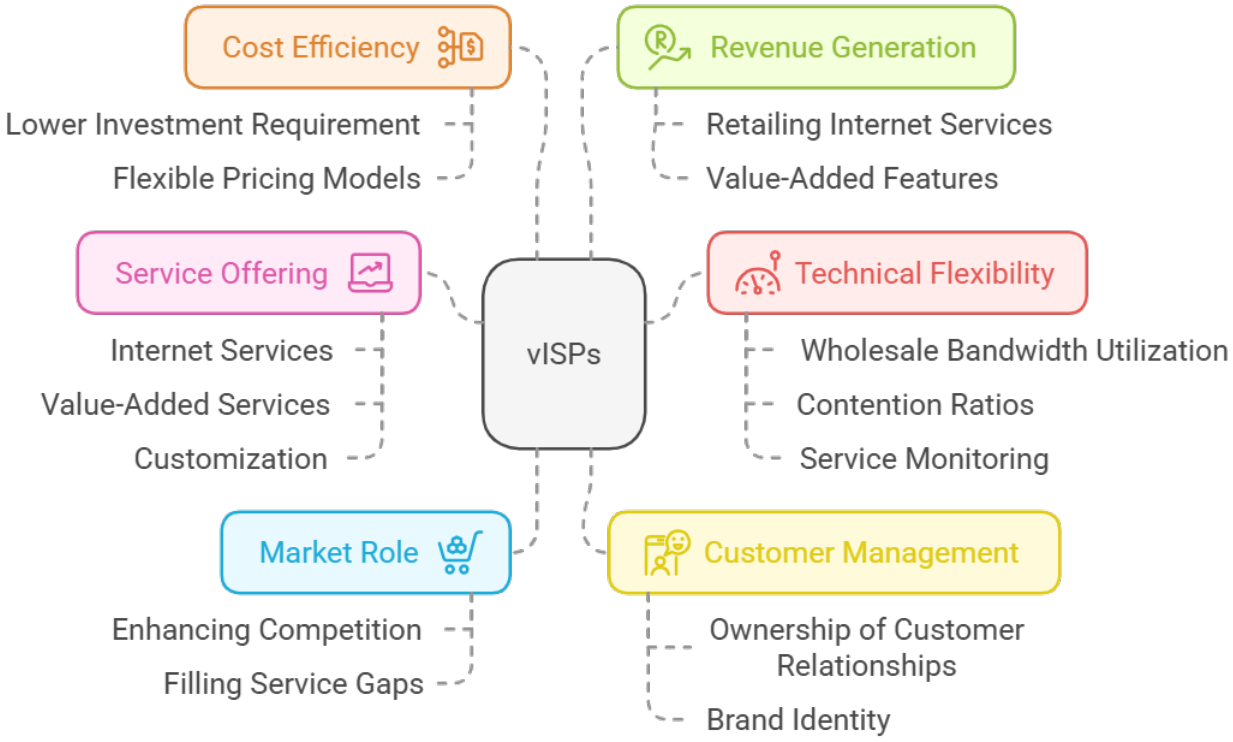

Figure 2.5: Key roles and benefits of vISPs

2.1.3 Challenges of vISPs

vISPs in Ethiopia face significant challenges caused by regulatory, operational, and market dynamics. Firstly, they need high startup capital and operational costs associated with constructing and maintaining essential infrastructure, encompassing access devices and service platforms to run the business. Secondly, vISPs are heavily reliant on wholesale internet services from primary providers, rendering them susceptible to disruptions or issues with the wholesale service (outages, delays in service provisioning, and maintenance support,) which directly impact service delivery and compromise reliability. Society lacks confidence in the private service providers on their legality and guesses that they are more

business-oriented than socially responsible. Thirdly, they have limitations on infrastructural coverage, especially in remote areas. vISPs focus on Addis Ababa areas that have better access and coverage than the regional parts. As vISPs don't have info on ethio telecom's infrastructure and expansion plan, to enable vISPs to focus on areas with high market demand but not addressed by ethio telecom. Also, the other challenges for vISPs are when broadening geographical coverage to cover rural and underserved communities, as it poses operational and cost challenges, which will limit their ability to increase the coverage of their services. The intense competition in the market serves as another major obstacle for vISPs, as direct rivalry with primary service providers makes securing market share difficult [15].

2.2 QoS and QoE in Influencing Factors in vISPs

2.2.1 QoS Factors

- **Service Up Time:** Refers to the amount of time during which a particular service, system, or application is operational and available for use. Often expressed as a percentage, it represents the service's availability and dependability over a given time period [11].

- **Delay:** Intrinsic to communications, it represents the time information consumes to reach the other side. Also referred to as latency, delay time can be increased if packets face long queues in the network (congestion) or take a less direct route to avoid congestion. The delay can be measured either one-way (total time from the

source that sends a packet to the destination that will receive it) or round-trip (one-way latency from source to destination plus the one-way latency from the destination back to the source) [11].

- **Packet Loss:** Occurs when one or more packets of data being transported across the internet or a computer network fail to reach their destination. Wireless and IP networks cannot guarantee that packets will be delivered, and some may be dropped if they arrive when their buffers are already full. Packet loss is the result of factors such as signal degradation, high loads on network links, corrupted packets being discarded, or defects in network elements [11].

- **Jitter:** The variation in delay caused by the variable transmission of the packets over the network. This can occur because of routers' internal queue behavior in certain circumstances (e.g., flow congestion), routing changes, etc. This parameter can seriously affect the quality of streaming audio and/or video [11].

- **Download Speed:** Signifies the pace at which data is transmitted from the internet to the device that is currently connected. It characterizes the rate of information transfer, delineating how swiftly content travels through the online network and reaches your specific device. Measured in bits per second (bps), kilobits per second (Kbps), megabits per second (Mbps), or gigabits per second (Gbps), depending on the scale of the transfer [11].

- **Upload Speed:** The opposite of download speed, it is the speed at which information travels from your internet-connected device to the internet. Upload speed is beneficial

for activities such as uploading files, sending emails with attachments, video conferencing, online gaming, and other tasks that require sending data from your device to the internet [11].

2.2.2 QoE Factors

The QoE influence factors are classified into three main categories: Human Influence Factors, System Influence Factors, and Context Influence Factors.

- **User Influence Factors:** encompass characteristics of users such as socioeconomic and demographic backgrounds, physical and mental states, emotional responses, motivation, and expectations, including attributes like age, gender, education level, prior experience, and preferences [1].

- **System Influence Factors:** relate to properties of the service or application that affect quality, including media-related parameters like encoding, resolution, and frame rate; network-related parameters such as bandwidth, delay, jitter, and packet loss; and device-related factors like device capability and configuration [1].

- **Context Influence Factors:** pertain to situational environmental aspects, including physical location, environmental noise, lighting conditions, timing and duration of use, social interactions, economic considerations like cost and subscription type, and task-related factors such as multitasking or parallel activities [1].

Finally, these factors collectively influence the perceived QoE by users, highlighting the multidimensional nature of QoE assessment.

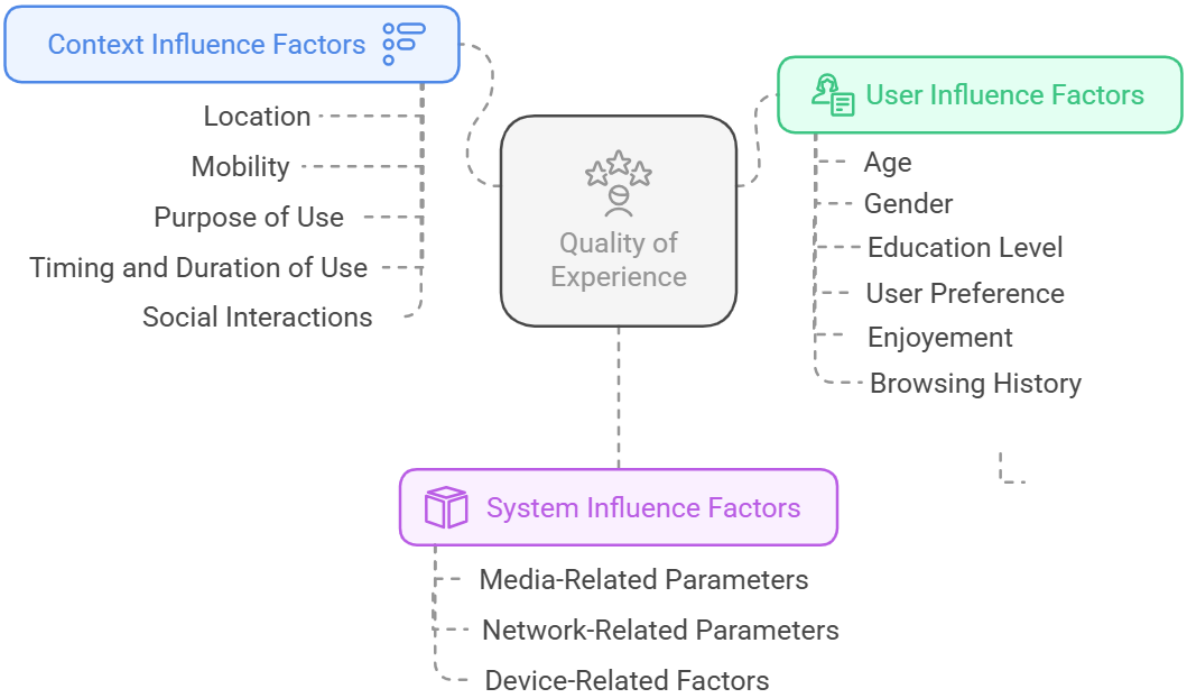

Figure 2.6: QoE influence factors

2.2.3 QoE Modeling Techniques

To increase the understanding of the results, the model performance analyses were carried out individually using technical data and a composite of the two (technical and subjective data).

It is quite challenging to measure the true user satisfaction in network services due to the inherently subjective and context-dependent nature of QoE. QoS, which comprises easily measurable parameters such as latency, jitter, and packet loss, QoE reflects individual perception, expectations, and emotions—elements that are not directly observable. The traditional approach to capturing QoE involves user surveys, which are costly and time-consuming, and resource-intensive, especially for large-scale or real-time applications. To overcome these limitations, mapping QoS to QoE offers a practical solution: by continuously

collecting QoS metrics from vISP systems, we can estimate the user's perceived experience without direct input, offering a scalable and resource-efficient path to QoE evaluation.

Several mapping techniques have been explored to realize this goal. Linear mapping was the earliest attempt, assuming a straightforward relationship between QoS and QoE. While computationally light, these models often fail to capture the nonlinear and multifactorial dynamics of user perception. On the other hand, machine learning is used more nowadays because it brings a big advantage: they don't assume a fixed or simple relationship between QoS and QoE. Linear models are fine when the relationship is straightforward, but in real life, user experience depends on a multitude of interdependent variables and varies depending on circumstances. Machine learning is capable of dealing with such complexity. These models can learn from data, adapt to patterns, and capture subtle, nonlinear connections that linear methods would miss. This makes them far more reliable when it comes to predicting how users actually feel about their service quality, especially in dynamic and varied network environments like those of virtual ISPs.

1. **Mapping QoS (Technical Data) to MOS value:**

   This approach estimates QoE by predicting MOS values directly from quantifiable technical QoS metrics. This computational modeling of QoE offers a way to evaluate service or application quality based on inherent performance characteristics such as bandwidth, latency, jitter, packet loss, average UL and DL speed.

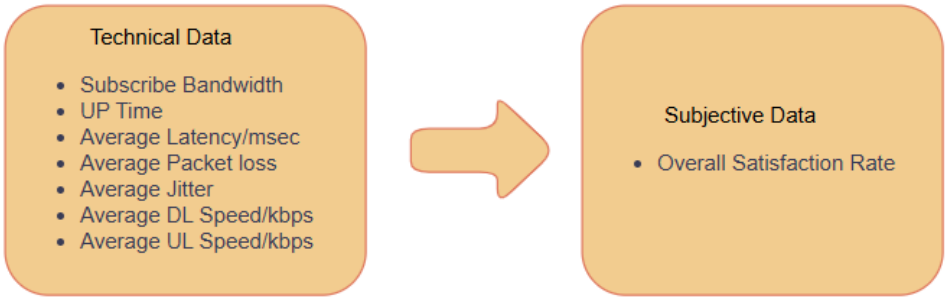

Figure 2.7: Technical data to MOS value mapping technique

2. **Mapping QoS (Technical Data) plus QoE (Subjective Data) to MOS Value:**

   This QoE modeling strategy integrates objective QoS metrics and subjective QoE parameters in a unified modeling strategy along with machine learning, network monitoring, contextual information, and adaptive streaming techniques. This comprehensive method will provide a more complete and accurate understanding of user experience and optimize service delivery.

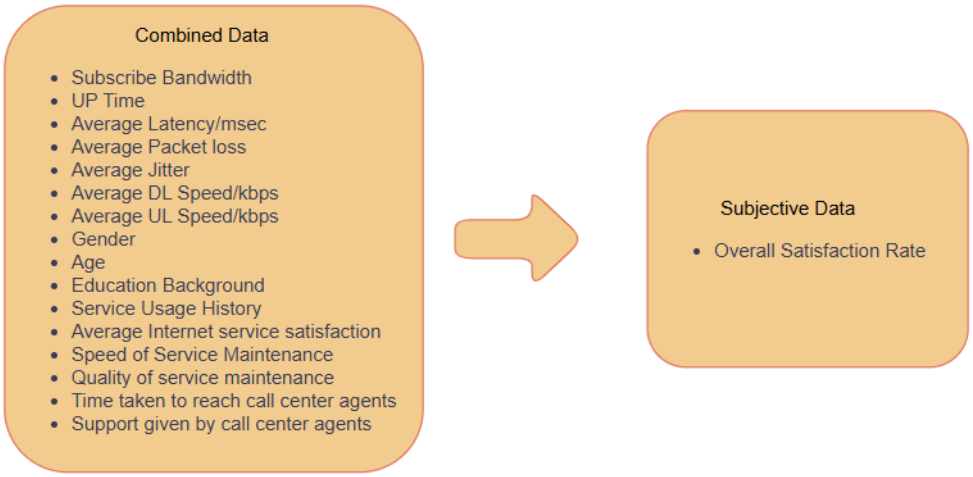

Figure 2.8: Composite of both data to MOS mapping technique

3. **Mapping QoS (Technical Data) to QoE (Subjective Data):**

   The approach focuses on modelling the user's perceived QoE, which is expressed through subjective assessment directly from quantitative technical QoS. This mapping aims to establish a direct relationship between network performance parameters and

the user's reported experience, which leads to enabling the estimation of QoE without requiring real-time subjective testing. This capability is valuable for proactive network management, service optimization, and understanding how technical impairments translate into user dissatisfaction [11].

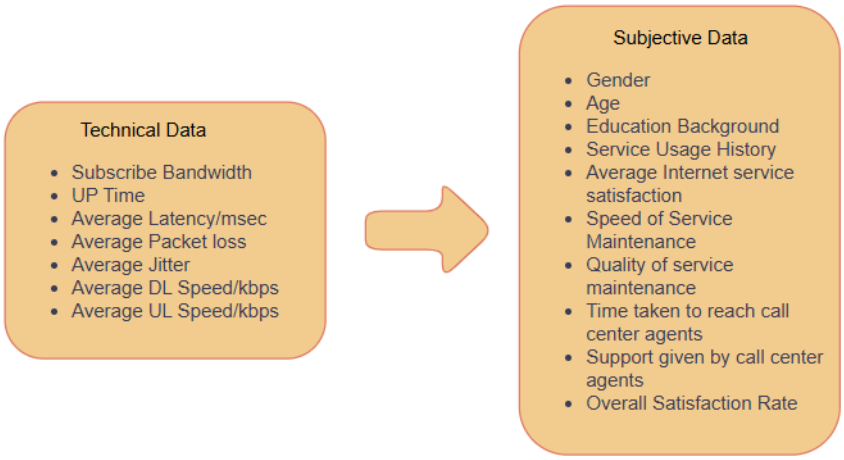

Figure 2.9: Technical to subjective data mapping technique

Chapter 3: Machine Learning for QoS to QoE Mapping

This chapter provides a comprehensive overview of various machine learning models for the specific task of QoS to QoE mapping in order to understand and predict user-perceived QoE. It begins by investigating HMMs about their components, the three fundamental problems with their common solutions, and applications. Then, followed by discussing different types, kernel functions, and hyperparameter tuning of SVMs. Finally, it covers learning techniques and hyperparameter considerations of RF algorithm.

3.1 Hidden Markov Model

3.1.1 Markov Chain

Markov Chains were introduced in 1906 by Andrei Andreyevich Markov (1856–1922) and named in his honor [17]. Markov chain, which is a stochastic modeling tool that is used in several fields of science and engineering to model real-world and randomly varying systems, processes, or phenomena. A Markov chain assumes that real-life systems are made of states (situations, outputs, or sets of values of a random variable at a given time t). MC assumes that a future state or output of a system depends on its present state only, and not on past iterated states. This dependency on the current state only

is sometimes characterized as "memorylessness" or Markov property (MP). This Markov chain is sometimes called first-order Markov chain. With knowledge of the future state, MC iteratively estimates the states of the system after a certain number of transitions in the future. In doing so, Markov models are used to understand the dynamics of the system, as the system transitions from one state to another. MC specifically refers to discrete-time Markov processes, and their study includes understanding transitions between states, periodicity, recurrence, and long-term behavior. One of the interesting things about systems that obey MC is that, from whichever initial states the chain starts from, the chain will converge to an equilibrium or steady state after sufficiently large iterations/transitions. In steady state, the system does not stop from transitioning; rather, from a given state, the probability of outward transitions and inward transitions will be the same (or is in equilibrium) so that the probability distribution stays the same.

Chain's rule

In probability theory, the chain rule (also called the general product rule) permits the calculation of any member of joint distribution of a set of random variables using only conditional probabilities [21]. Suppose that we have a set of random variables (observations) $q_1, q_2, ...q_i, ..., q_h$, where $q_i \in s_1, s_2, ..., s_N$. Then, the probability of occurrence of the observations is:

$$\begin{aligned} P(q_1, q_2, ..., q_h) &= P(q_1)P(q_2|q_1)P(q_3|q_2, q_1)...P(q_h|q_{h-1}, ..., q_1) \\ &= P(q_1)\prod_{i=2}^{h}(P(q_i|_{i-1}, ..., q_2, q_1) \end{aligned} \tag{3.1}$$

The following Figure 3.1 shows a visualization of a Markov model with two states connected by four transitions, which helps us to understand these concepts [23].

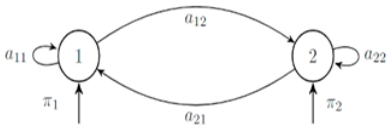

Figure 3.1: Discrete-time Markov Chain

In Figure 3.1, the states are represented by circles numbered in it. The probabilities of getting from one state to the other are called transition probabilities. For example, the transition probabilities between states in the above figure are assigned by a_{ij}. The general transition probability matrix for N hidden states would be:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}_{NXN} \tag{3.2}$$

Where N is an integer that is found in $1 \leq n \leq N$ and it represents the size of the transition matrix, which is the maximum number of states under study. Each elements (a_{ij}) in the above probability matrix tells us the probabilities of staying in state i at time t or the transition probabilities from state i to j at time t + 1. Moreover, the transition probabilities for a given state S_i should satisfy one condition:

$$\sum_{j=0}^{N} (a_{ij}) = 1 \tag{3.3}$$

The variable π_i in the Figure 3.1 represents the probability of starting at state i. The

general representation of the initial distribution of MC for N states would be:

$$\pi = \begin{bmatrix} \pi_1 = P(q_1 = 1) \\ \pi_2 = P(q_2 = 2) \\ \vdots \\ \pi_N = P(q_{nb} = N) \end{bmatrix}_{NX1} \tag{3.4}$$

Where N is the number of states under the study. The requirement is that the total probabilities of starting states must satisfy a condition:

$$\sum_{i=0}^{N} (\pi_i) = 1 \tag{3.5}$$

Lastly, if all states x_t are known or observable, the output will be the sequence of the states from an initial to a final state, and we will call this sequence the observation sequence O. The calculation for probability of the observed sequence would be:

$$\begin{aligned} P(O = Q|A, \pi) &= P(q_1) \prod_{t=2}^{T} (P(q_t|q_{t-1}) \\ &= \pi_{q1} a_{q_1 q_2} ... a_{q_{T-1} q_T} \end{aligned} \tag{3.6}$$

Where π represents the sum of initial probabilities and A is the vector of transition probabilities. π_{q1} represents the probability of starting at the state q_1. While $a_{q_1 q_2}$ represents the probability of reaching state q_2 from q_1. Equations 3.2 to 3.6 describe the properties of a first-order Markov process; hence, A and π are typical parameters that describe MC.

3.1.2 Hidden Markov Models

One inherent assumption in Markov chain include: the random variables, resulting from state transitions, are observable and transition probabilities can be computed directly from the observation. An MC is useful when we need to compute a probability for a sequence of observable events. In many real-world problems, however, the states we are interested in are hidden or nonobservable: we do not observe them directly, but their influence on the observable events can be inferred. A Hidden Markov Model (HMM) allows us to talk about both observed events and hidden events that we think of as causal factors in our probabilistic model. HMMs are probabilistic models that are applied when states are not directly observable or when the states are hidden, underlying process that transitions between a number of states, the model assumes the presence of other observable parameters (emissions) that are related to the hidden states with some probability function. The observable parameters or symbols are also called emissions. The hidden process has a finite state space and is assumed to be a Markovian process, i.e., it is governed by a Markov chain [21].

HMM is applied when the states are not directly observable or are hidden. The model assumes that the observable emissions are related to or "influenced by" the outcomes of the hidden states in a known way by some probability function. Or transitions of the hidden states cause transitions in time of the emission states; hence, we note two transitions in time by the hidden and emission states. This way, HMM can be conceptualized as a kind of double stochastic process, where the first is the process within the hidden states, while the second is the emission process.

3.1.3 Components of HMM parameter

An HMM is specified by the following components:

1. A set of hidden states:

   These are the unobservable states that generate the observed data.

$$Q = \{q_1, q_2, ..., q_N\} \tag{3.7}$$

   Where, N is the number of hidden states.

2. A transition probability matrix: shows all possible state transitions. Entries of the matrix are transition probabilities from a given state to all other states, including the probability of staying in the same state. The entries

$$A = \{a_{11}, a_{ij}, ..., a_{NN}\} \tag{3.8}$$

   Where, each a_{ij} represents the probability of moving from state i to state j.

3. An initial probability distribution: is usually expressed as a probability distribution vector, whose entries indicate the probability that the system will be in a given state at a given initial time.

$$\pi = \{\pi_1, \pi_2, ..., \pi_N\} \tag{3.9}$$

   Where, π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states.

4. A set of observation sequence:

   These are the directly measurable data points.

$$O = \{o_1, o_2, ..., o_M\} \tag{3.10}$$

   Where, M is the number of observable sequences.

5. a sequence of observation likelihoods (emission probabilities):

$$B = \{b_i(o_t)\} \tag{3.11}$$

   Where each expresses the probability of an observation o_t being generated from a state q_i.

   The observation probability, also known as the emission probability B usually represented as a matrix specifying the probability of observing a particular symbol or output given a certain hidden state. The rows in the matrix show the number of hidden states, and the columns of the matrix show observation symbols. Here is the general emission matrix:

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix}_{NXM} \tag{3.12}$$

   Where N represents the maximum number of hidden states and M represents the maximum length of observation symbols. Hence, N and M determine the size of the

emission probability matrix.

Thus, N, A, π, M, and B are the five crucial components of HMM. The compact form of the five HMM elements, which are a standardized and widely used HMM parameter, is denoted by:

$$\lambda = (A, B, \pi) \tag{3.13}$$

These parameters define the underlying probabilistic structure of HMM. By computing these parameters from a set of training data, we can observe how the system operates and use it to make predictions or decisions in applications.

Representing and visualizing the entire HMM process is a bit complex because of the addition of the time dimension on top of the transitions in the hidden states and emissions. Assume that we have a system with N hidden and M emission states.

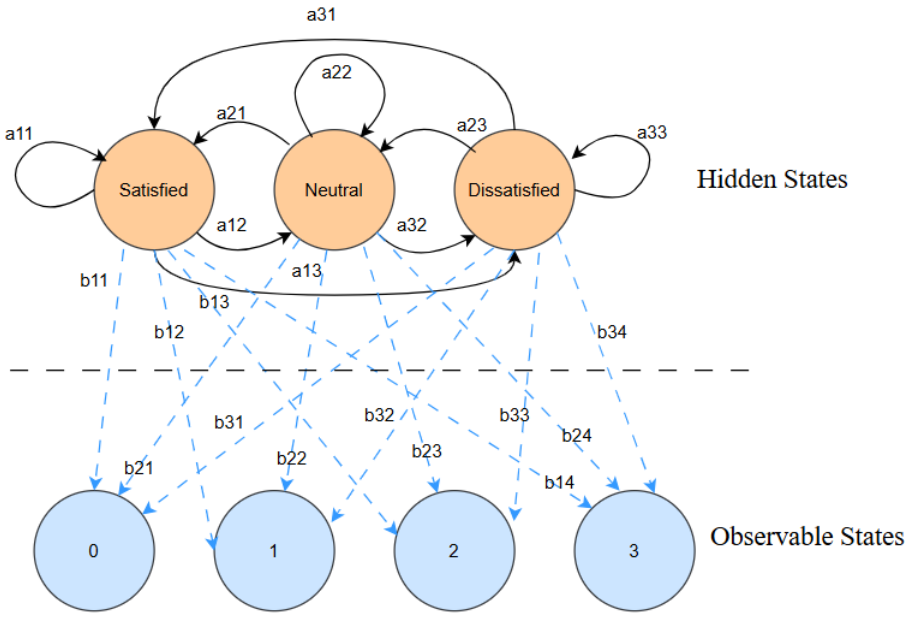

Figure 3.2: Hidden Markov Model

The illustration in Figure 3.2 describes the structure of HMM that contains all the fundamental components listed above, where the orange circles labeled as ("Satisfied," "Neutral," and "Dissatisfied") represent hidden state,s which is the underlying, unobservable states of the system we are trying to infer and in this thesis context, these represent user satisfaction that we cannot directly measure but aim to predict. The blue circles labeled as ("0," "1," "2," and "3") represent observable states. The solid black arrows connecting the hidden states represent the hidden states transitions at different times, called the transition Probabilities (a_{ij}). The transitions are from each state to every other state in the next instant, and the state transition matrix captures these transitions. a_{13} indicates the probability of transitioning from the satisfied hidden state to the Dissatisfied state in a single time step. Also, a_{11}, a_{22}, and a_{33} indicate the probability of the states staying in the same state at the next time step. These transition probabilities define the dynamics of how the hidden states evolve. The broken blue arrows originating from each hidden state and pointing towards the observable states represent the emission probabilities denoted as (b_{ij}) and show the interactions among the hidden and emission states at a given time t. Note that each emission state is influenced (in a probabilistic sense) by all hidden states, which are captured via the emission matrix. b_{11} signifies the probability of observing the observable state 0 when the underlying hidden state is satisfied. These emission probabilities define the likelihood of observing a particular piece of observable data given the current hidden state.

In summary, Figure 3.2 shows the transitions observed in the hidden states, interaction

among the hidden and emission states, and the transitions of both states at different time instants.

3.1.4 Fundamental Problems in HMM

An influential tutorial by Rabiner (1989), based on tutorials by Jack Ferguson in the 1960s, introduced the idea that hidden Markov models should be characterized by three fundamental problems that we can solve using HMMs [19].

Problem 1 (Likelihood)

Given the model (A, B, π) and a sequence of observations O, find P(O | λ). Here, we want to determine a score for the observed sequence O with respect to the given model λ [20].

Problem 2 (Decoding)

Given the model (A, B, π) and an observation sequence O, find an optimal state sequence for the underlying Markov process. In other words, we want to uncover the hidden part of the Hidden Markov Model [20].

Problem 3 (Learning)

Given an observation sequence O and the dimensions N and M, find the model (A, B, π) that maximizes the probability of O. This can be viewed as training a model to best fit the observed data. Alternatively, we can view this as a (discrete) hill climb on the parameter space represented by A, B and λ [20].

3.1.5 Common Solutions to HMM Problems

Likelihood Computation: The Forward Algorithm

The forward algorithm is a kind algorithm of dynamic programming algorithm, that is, an algorithm that uses a table to store intermediate values as it builds up the probability of the observation sequence. The forward algorithm computes the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence, but it does so efficiently by implicitly folding each of these paths into a single forward trellis [19].

$$P(O|\lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i) \tag{3.14}$$

For an HMM with N hidden states and an observation sequence of T observa tions, there are NT possible hidden sequences. For real tasks, where N and T are both large, NT is a very large number, so we cannot compute the total observation likelihood by computing a separate observation likelihood for each hidden state sequence and then summing them. Instead of using such an extremely exponential algorithm, we use an efficient $O(N^2T)$ algorithm.

Backward Algorithm

Computes the probability of the partial observation sequence from time t+1 to the end, given that the system is in state i at time t.

The Backward Algorithm is a key component of HMMs, a statistical framework used

to model sequences of observations. Unlike the Forward Algorithm, which calculates the probability of being in a particular state at a specific time given the observations up to that point, the Backward Algorithm calculates the probability of observing the remaining sequence of observations given that the model is in a specific state at that time.

The Backward Algorithm essentially works backward through the sequence of observations, calculating the probability of observing the remaining sequence given the current state at each time step.

Decoding: The Viterbi Algorithm

The Viterbi algorithm was first proposed by Andrew Viterbi in 1967 and is named after the algorithm's inventor. Viterbi is a kind of dynamic programming algorithm that makes use of a dynamic programming trellis and finds the most probable sequence of hidden states that generate a given sequence of observations [19]. The Viterbi algorithm is similar to the forward algorithm. The main difference is that the forward algorithm uses the sum over previous states, whereas the Viterbi algorithm uses maximization [23]. Given a series of observed events, the Viterbi algorithm efficiently determines the most likely sequence of hidden states in an HMM, called the Viterbi path, which results in a sequence of observed events, dividing the issue into smaller subproblems and solving them recursively.

The Viterbi algorithm operates by continuously computing the most likely paths to all states at each time step, storing the probabilities, and backtracking the most probable ones to obtain the most likely hidden state sequence. The algorithm ensures efficient and exact decoding of hidden state sequences; thereby, it is essential in applications where there is a requirement for precise sequence analysis and pattern recognition. The algorithm has

been extensively applied in the convolution code decoding, speech recognition applications, keyword spotting, computational linguistics, and bioinformatics [24].

To find the best single hidden state sequence S = $(q_1, q_2, ..., q_T)$ for the given observation sequence O = (o_1, o_2, ..., o_T) and model λ. The probability of being in state i at time t given the observation sequence O and the model λ is given by [25]:

$$\begin{aligned} \gamma_t(i) &= P(q_t = i|O, \lambda) \\ &= \frac{P(O, q_t = i|\lambda)}{P(O|\lambda)} \\ &= \frac{P(O, q_t = i|\lambda)}{\sum_{i=1}^{N} P(O, q_t = i|\lambda)} \end{aligned} \tag{3.15}$$

$P(O, q_t = i|\lambda)$ can be found as the joint probability. Hence, the above equation can be rewritten as:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{3.16}$$

HMM Training: The Baum-Welch Algorithm

The most common approach to parameter estimation for HMMs is the Expectation Maximization (EM) algorithm. The Baum-Welch algorithm is the EM algorithm for HMMs. Baum-Welch algorithm is used to train the parameters of an HMM, the A transition probability matrix and the B observation likelihood matrix [19]. The name "Expectation-Maximization" comes from the fact that every iteration in the algorithm has an expectation step (E-step) followed by a maximization step (M-step). The algorithm starts with an log-likelihood function estimate of the model parameters for the full data set (observations and

missing data). The conditional log-likelihood is then maximized (M-step) with respect to model parameters after the conditional expectation of the log-likelihood is calculated (E-step) given the observations. The next iteration uses the estimated values of the parameters that were obtained, and these are updated until either the number of iterations is reached or some convergence criteria are satisfied [29].

Given the γ and "di-gamma" we verify that the model λ = (A, B, π) can be re-estimated as follows:

1. For i = 0, 1,...,N-1, let $\pi_i = \gamma_o$(i);

2. For i = 0, 1,...,N-1 and j = 0, 1,...,N-1, compute:

$$a_{ij} = \frac{\sum_{t=0}^{T-2} \gamma_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)} \tag{3.17}$$

3. For i = 0, 1,...,N-1 and k = 0, 1,...,M-1, compute:

$$b_j(k) = \frac{\sum_{t\epsilon\{0,1,...,T-1;O_t=k\}} \gamma_t(j)}{\sum_{t=0}^{T-1} \gamma_t(j)} \tag{3.18}$$

The numerator of the re-estimated a_{ij} can be seen to give the expected number of transitions from state q_i to state q_j, while the denominator is the expected number of transitions from q_i to any state. Then the ratio is the probability of transiting from state q_i to state q_j, which is the desired value of a_{ij}. The numerator of the re-estimated b_j(k) is the expected number of times the model is in state q_j with observation k, while the denominator is the expected number of times the model is in state q_j. The ratio is the probability of observing symbol k, given that the model is in state q_j, which is the desired value of b_j(k) [20, 28].

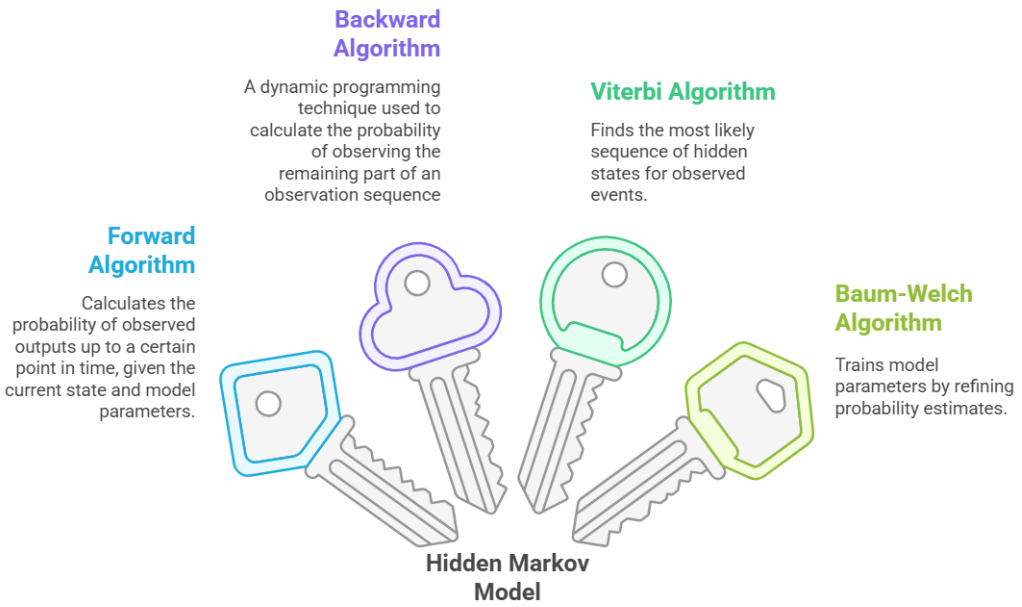

Figure 3.3: Key HMM algorithms

3.1.6 Applications of HMMs

Here are a wide range of applications of HMMs:

- Dynamic Spectrum Allocation and Channel Prediction in Cognitive Radio: used HMMs to model and predict the spectrum occupancy of licensed radio bands that can dynamically select different licensed bands for use with significantly less interference from and to the licensed users. By predicting the duration of spectrum holes of primary users, the CR can utilize them more efficiently by leaving the band that it currently occupies before the start of traffic from the primary user of that band [21].

- Signal Processing: HMMs are used in signal processing for speaker identification in voice and audio by modeling speaker-specific features as hidden states [22].

- Voice Recognition- HMMs are used to represent phonemes and words in audio data by Automatic Speech Recognition (ASR) systems like Google's voice recognition.

Further, Part-of-speech tagging is a typical Natural Language Processing (NLP) activity that uses HMMs, which aids in syntactic analysis, improving machine comprehension of text, and assisting machine translation systems by classifying the components of speech (such as nouns, verbs, and adjectives) associated with each word in a phrase [22].

- Traffic Prediction in Next Generation Mobile Network: The QoS parameters for the wireless networks are predicted using the HMM model. Using an HMM to describe packet-level network traffic for fixed and variable packet sizes. Since it can model hidden states like user behavior while learning the traffic characteristics from the accessible traces, HMM-based traffic modeling has become increasingly popular [21].

- Bioinformatics: HMMs have been used for the bioinformatics problem of gene prediction to locate coding areas in DNA sequences. The annotation of genomes is aided by the ability of HMMs to distinguish between coding and non-coding sections, or exons and introns [22].

- Channel classification in mobile networks: By constructing a channel estimation tool based on hidden Markov models to estimate the transmission channel characteristics in mobile radio receivers.

- Robotics: HMMs have been used for the robotics problem of Simultaneous Localization And Mapping (SLAM) in autonomous systems. Robots utilize HMMs to analyze sensor data, such as lidar or camera inputs, to concurrently estimate their self-location and build maps of their surroundings [22].

3.2 Support Vector Machines

Support Vector Machines (SVMs) originated in the 1990s and were presented as powerful supervised learning algorithms grounded in statistical learning theory, effectively bridging the gap between theoretical analysis ability and practical application in pattern recognition and regression. SVMs excel at classifying data by constructing models from labeled training examples to predict the category of new instances, demonstrating a strong ability to generalize across diverse real-world problems like image and speech recognition, text categorization, and anomaly detection, making them a valuable tool for data separation in various fields.

SVMs are primarily used for pattern classification, dealing with both linear and Non-linear separable patterns, while linear patterns can be easily distinguished in lower dimensions, non-linear patterns require further manipulation for effective separation. The core concept of SVM for linearly separable patterns is to construct an optimal hyperplane, which is a decision boundary chosen from many possible hyperplanes, that maximizes the margin. The margin is the distance between the hyperplane and each pattern's nearest data points. Since a larger margin typically results in a more accurate classification of the given patterns, SVM's main objective is to achieve the largest possible margin.

The equation shown below is the hyperplane representation:

$$Hyperplane, aX + bY = c$$

The best hyperplane is the plane that has the maximum distance from both classes,

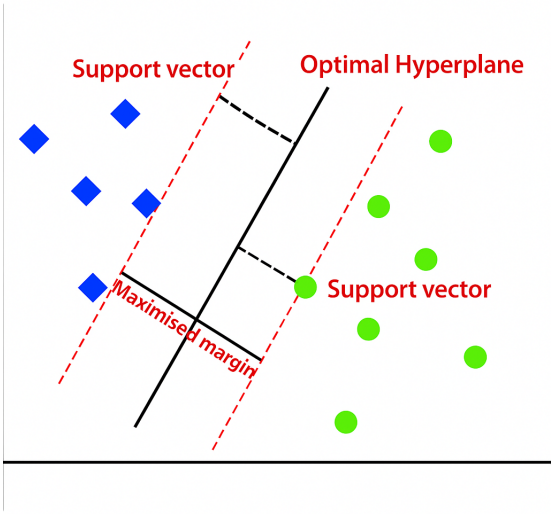

Figure 3.4: SVM hyperplane [41]

and this is the main aim of SVM. This is done by finding different hyperplanes that classify the labels in the best way, then it will choose the one that is farthest from the data points or the one that has a maximum margin.

3.2.1 Types of SVMs

Linear SVM

Linear SVMs classify linearly separable data, meaning the data can be divided into clearly noticeable classes by a single straight line in two dimensions, or a hyperplane in higher dimensions. A linear SVM identifies this optimal boundary, called a hyperplane, between two classes, aiming to maximize the margin. Margin is the distance between the hyperplane and the closest data points of each class. Assuming that the data is linearly separable in the input space, this maximization improves the SVM's capacity for generalization Figure 3.5. This indicates that the data points can be represented so that they can be efficiently separated by a linear function.

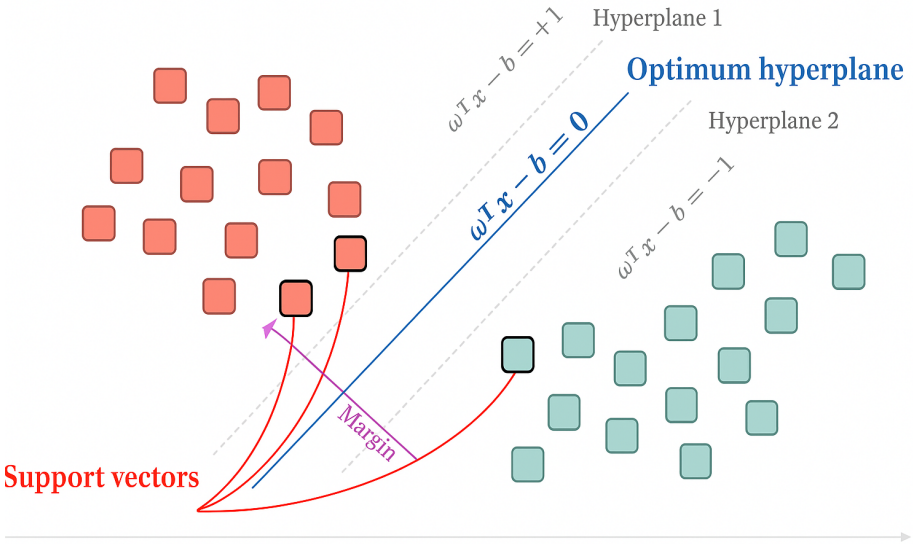

Figure 3.5: SVM example of linearly separable data [31]

Non-Linear SVM

Non-linear SVM can be used to classify data that is not linear, meaning that it cannot be separated into two classes by a straight line in the case of 2D. Non-linear SVMs employ kernel functions in order to convert the input data into a higher-dimensional space where the data points can be more easily separated by a hyperplane. This transformation makes SVMs capable of handling data sets where the classes are not easily distinguishable with a linear boundary in the original feature space. A key component of SVMs is the "kernel trick," which allows them to classify complex datasets that linear SVMs are unable to handle by modeling complicated connections throughout the data. The kernel trick allows SVMs to map the data to a higher dimension and, hence, allow the use of non-linear decision boundaries within the original space, which makes SVMs much more flexible and robust.

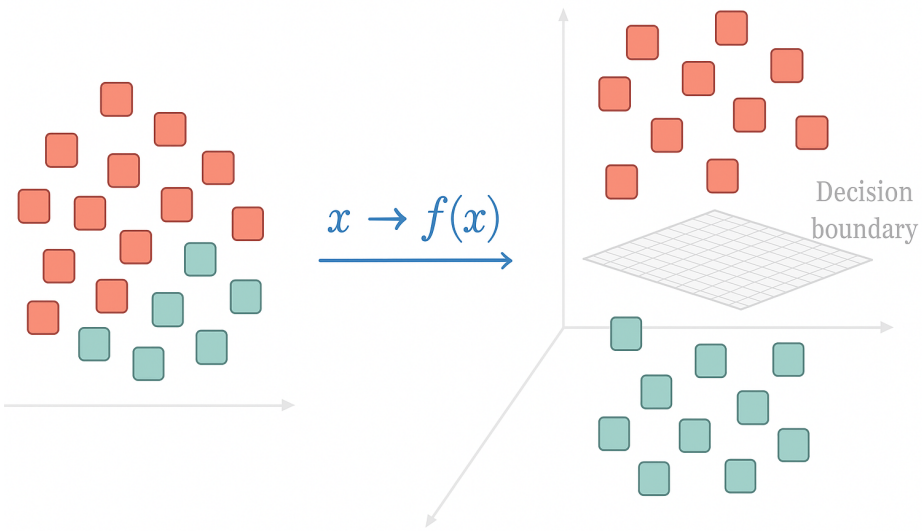

Figure 3.6: SVM example of nonlinearly separable data with the kernel trick [31]

3.2.2 Kernel Functions

The performance of SVM largely depends on the suitable selection of a kernel function that generates the dot products in the higher-dimensional feature space. This space can theoretically be of an infinite dimension where the linear discrimination is possible. There are several kernel models to build different SVMs:

Linear Kernel:

A linear kernel is the simplest form of kernel used in SVM. It is suitable when the data is linearly separable meaning that a straight line (or hyperplane in higher dimensions) can effectively separate the classes [41].

$$K(x, y) = x.y \tag{3.19}$$

Polynomial Kernel:

The polynomial kernel allows SVM to model more complex relationships by introducing polynomial terms. It is useful when the data is not linearly separable but still follows a pattern [41].

$$K(x, y) = (x.y + c)^d \tag{3.20}$$

Radial Basis Function (RBF):

The RBF kernel is the most widely used kernel in SVM. It maps the data into an infinite-dimensional space, making it highly effective for complex classification problems [41].

$$K(x, y) = e^{-}(\gamma||x–y||^2) \tag{3.21}$$

Sigmoid Kernel:

The sigmoid kernel is inspired by neural networks and behaves similarly to the activation function of a neuron. It is based on the hyperbolic tangent function and is suitable for neural networks and other non-linear classifiers [41].

$$K(x, y) = tanh(\gamma.x^T y + r) \tag{3.22}$$

3.2.3 SVM Hyperparameters

Optimizing SVM's performance requires selecting the appropriate values for C and γ because these hyperparameters influence the decision boundary and, consequently, the

accuracy and the model's ability to generalize.

C: Regularization Parameter: balances between maximizing the margin and minimizing the training error. The C parameter is inversely proportional to the margin size, meaning that the larger the value of C, the smaller the margin whereas the smaller the value of C, the larger the margin.[35].

- **Higher C:** A larger allows it to capture more complicated relationships within the data, which increases the model complexity and flexibility. However, being flexible makes the model more susceptible to overfitting and sensitive to noisy data.

- **Lower C:** A smaller C value gives priority to a wider margin, which helps to stop overfitting. Although if the margin becomes too large, it can lead to underfitting. Also, it can potentially sacrifice some accuracy on the training data.

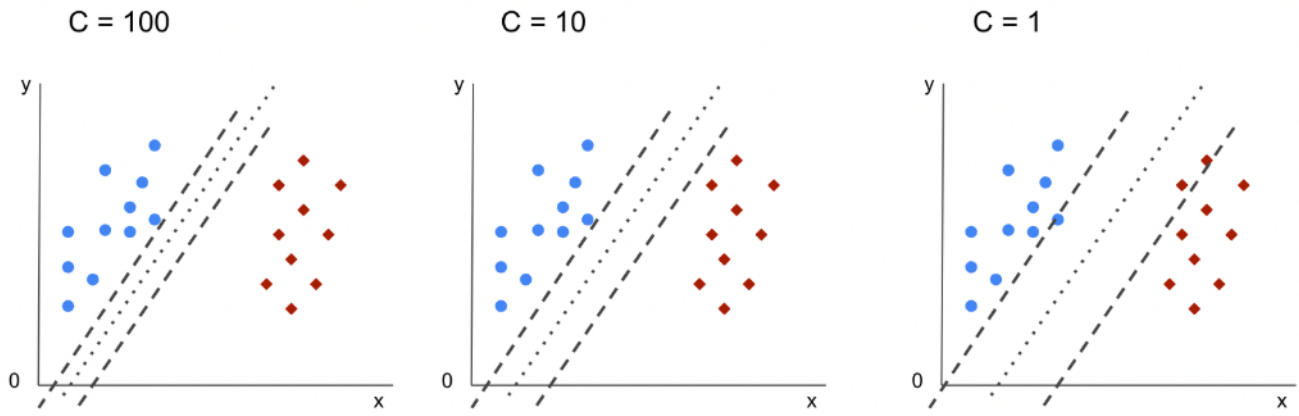

Figure 3.7: C hyperparameter [41]

Gamma: Gamma: It determines the impact on the decision boundary of each data point, because it influences the decision boundary's smoothness and shape.

- **Higher Gamma:** values increase model complexity and flexibility, allowing for more

complicate decision boundaries that capture subtle data relationships, but this can lead to overfitting and increased sensitivity to noise.

- **Lower Gamma:** values promote a simpler, more generalized model with a smoother decision boundary, reducing the risk of overfitting, though this may sacrifice the ability to model complex patterns.

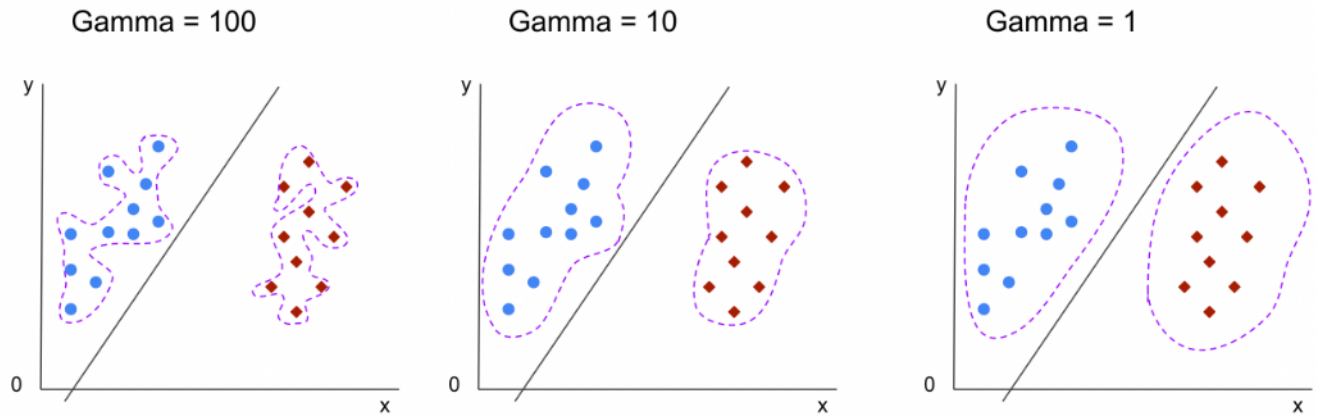

Figure 3.8: Gamma hyperparameter [41]

3.3 Random Forest Algorithm

Random Forest (RF) is an ensemble learning approach, developed by Breiman for solving classification and regression problems. Ensemble learning is a machine learning scheme to boost accuracy by integrating multiple models to solve the same problem. In particular, multiple classifiers participate in ensemble classification to obtain more accurate results compared to a single classifier. In other words, the integration of multiple classifiers decreases variance, especially in the case of unstable classifiers, and may produce more reliable results. Next, a voting scenario is designed to assign a label to unlabeled samples. The commonly used voting approach is majority voting, which assigns the label with the

maximum number of votes from various classifiers to each unlabeled sample. The popularity of the majority voting method is due to its simplicity and effectiveness. More advanced voting approaches, such as the veto voting method, in which a single classifier vetoes the choice of other classifiers, can be considered as an alternative to the majority voting method [31].

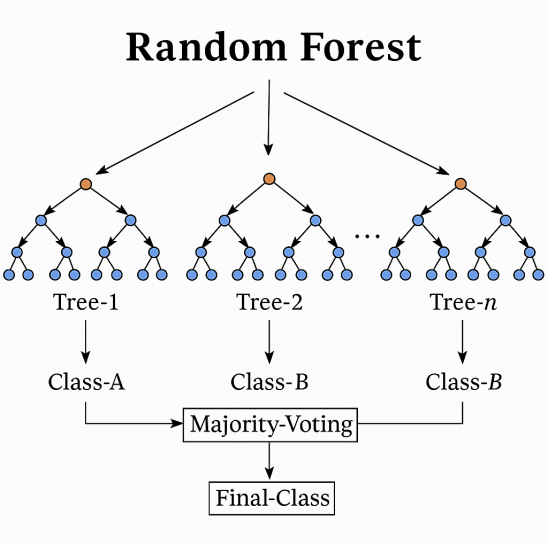

Figure 3.9: Random Forest Algorithm [40]

3.3.1 Random Forest Learning Techniques

Bagging (Bootstrap Aggregating)

Bagging, also known as Bootstrap Aggregating, is an ensemble learning technique intended to lower variance while increasing the accuracy and stability of the model. It achieves this by training multiple instances of the same learning algorithm on different subsets of the data and through a process called bootstrapping, by creating multiple subsets of the original training data. Since the original data is randomly sampled with replacement to create each of these subsets, then some data points may appear more than once in a single subset, while others may be left out. A base model, often a decision tree, is then trained

on each of these bootstrapped subsets, resulting in an ensemble of models. Bagging trains each model individually on a different subset of the data, and all the resulting models are more diverse. This diversity is then utilized to reduce the overall variance of the ensemble, making it less likely to overfit the training data.

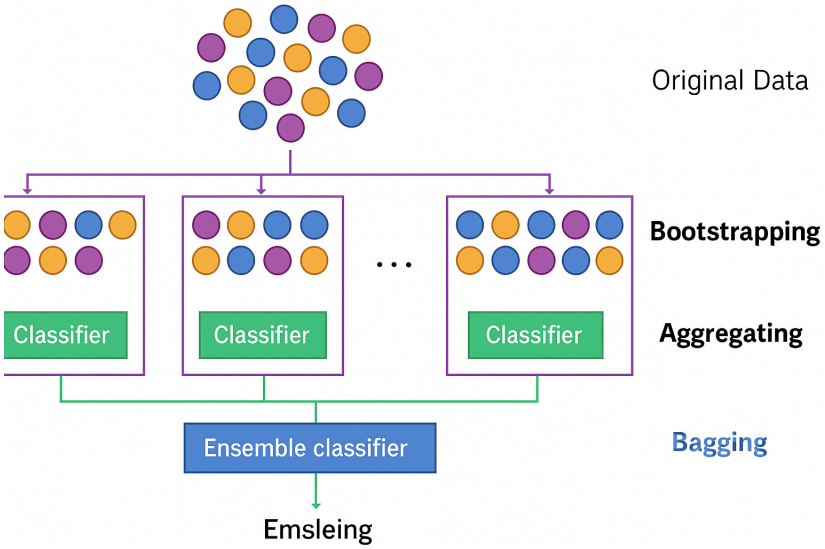

Figure 3.10: Bootstrap aggregating technique [42]

Boosting

Boosting is a process of building a sequence of models, where each model attempts to correct the error of the previous one in that sequence. AdaBoost was the first successful boosting approach, which was developed for binary classification cases. However, the main problem of AdaBoost is model overfitting.

Boosting is an ensemble learning method that works by sequentially training a series of models. The basic idea is to combine the strengths of multiple "weak learners" to create a powerful "strong learner." Each model in the sequence focuses on correcting the errors made by its predecessors. This is achieved by assigning weights to the training data, where the weights are adjusted after each model is trained. Instances that are misclassified by

one model receive higher weights, forcing subsequent models to prioritize them.

Here's a more detailed look at how boosting works:

1. Initialization: The algorithm begins by giving all training samples the same weights.

2. Model Training: A weak learner is trained on the training data then the algorithm evaluates the performance of the weak learner. Samples that were misclassified by the learner are assigned higher weights, while correctly classified samples are assigned lower weights. This step highlights the difficult-to-classify samples and proceeds to the next iteration, where a new weak learner is trained on the re-weighted data, concentrating on the samples that the previous learner had trouble with.

3. Combining Models: The process of training and re-weighting is repeated for a specified number of iterations. Finally, the predictions of all the weak learners are combined through a weighted voting or averaging process to make the final prediction. More proficient learners are given more weight while combining.

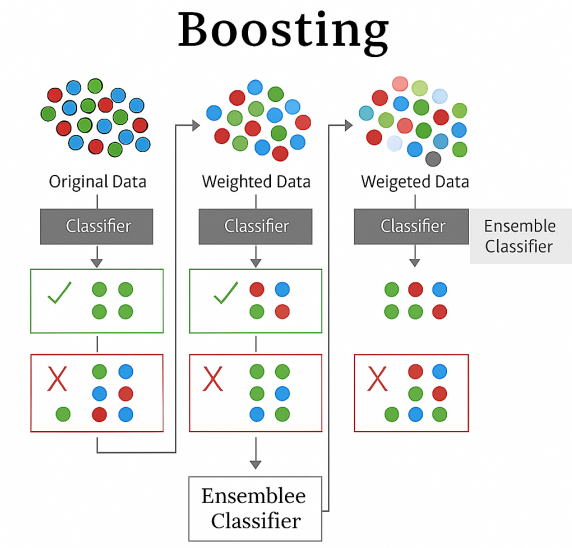

Figure 3.11: RF boosting technique [42]

3.3.2 Random Forest Hyperparameters

In order to ensure that the RF is optimized, the hyperparameters have to be set carefully [39]. Here are some of the various hyperparameters for RFs:

1. N_estimators: The number of trees in the Random Forest. Usually, the more trees, the better [40].

   Default = 100.

2. Max Depth: The maximum depth of each tree in the Random Forest. If this is not defined, the nodes in all trees will be expanded to the extent that all the leaf nodes are pure or all the leaf nodes are pure or until all leaf nodes contain less than min_samples_split [40].

   Default = None

3. Max Features: The number of features to consider when looking for the best split. For example, if there are 35 features in a dataframe and max_features is 9, only 9 of the 35 features will be used in the decision tree [40].

   Default = None

4. min_samples_split: It determines the minimum amount of data points which should be used to split a node. A larger value can prevent overfitting but might limit the model's ability to learn complex relationships [40].

5. min_samples_leaf: This defines the minimum number of data points allowed in a leaf node. A larger value can reduce overfitting, but might make the model too simple.

Chapter 4: Methods for QoS to QoE Mapping

This chapter outlines the systematic approach taken to develop a model for mapping QoS parameters into QoE metrics. It begins by defining the vISPs system model as the foundational framework and then presents the process, including data collection and data preprocessing stages. This preprocessing phase is further broken down into feature selection to identify relevant data attributes, such as hidden state formation to define underlying experiential states, and observation sequence formation to structure the data for model input.

4.1 vISPs System Model

The system model illustrated in the figure below shows in detail our approach to modeling QoE for ethio telecom, Websprix IT Solution PLC, and ZERGAW Cloud.

The two flowcharts below provide a visual representation of how we conceptualized the problem and the key components involved in the modeling process. It begins with data collection from the ISP and vISPs. The collected data is passed through a preprocessed process that involves data cleaning, normalization, and feature selection to ensure the data fits with the models. Figure 4.1 presents the QoE prediction system model using HMM.

After preprocessing of the data, the next step is to define the hidden states and observable states, and then data splitting is done. Before the training of data clustering is done on the data, because the data has multiple parameters, we need to convert it to one equivalent parameter. We used the Baum-Welch algorithm for model training to estimate the HMM parameters, including initial, transition, and emission probabilities, and then the Viterbi algorithm to find the most probable sequence of hidden states.

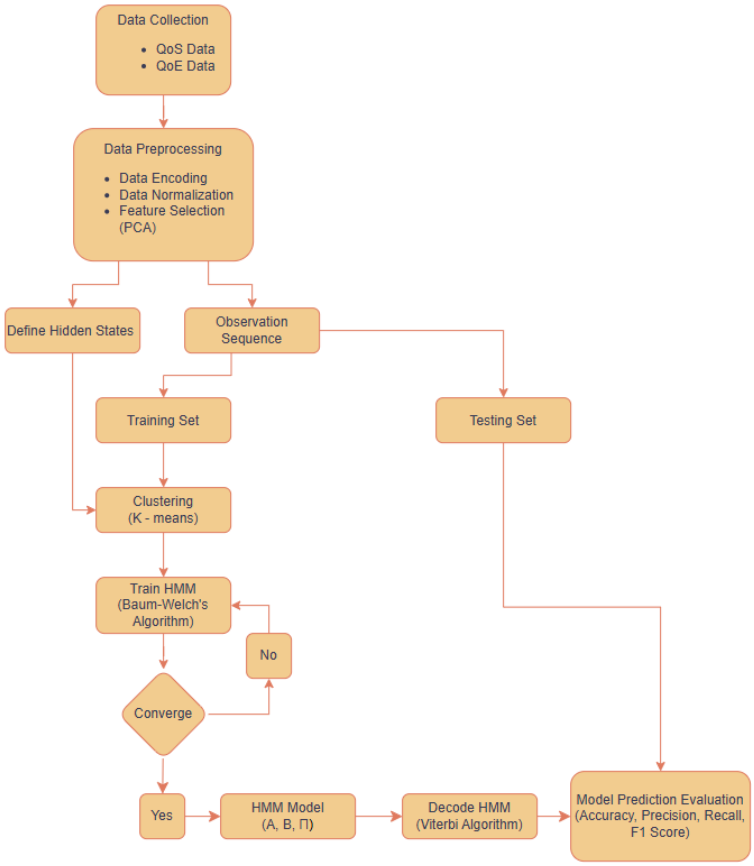

Figure 4.1: HMM system model for predicting QoE

Figure 4.2 system model presents QoE prediction using SVM and RF. It is the same as HMM until data preprocessing, but after data preprocessing grid search technique is used to choose the optimal kernel type, C, and Gamma parameter. SVM and RF models were trained using the selected features and hyperparameters. Once the models were trained

and parameters were obtained, we predicted the user experience and the sequence of the QoE over time. Finally, model performance evaluation was conducted.

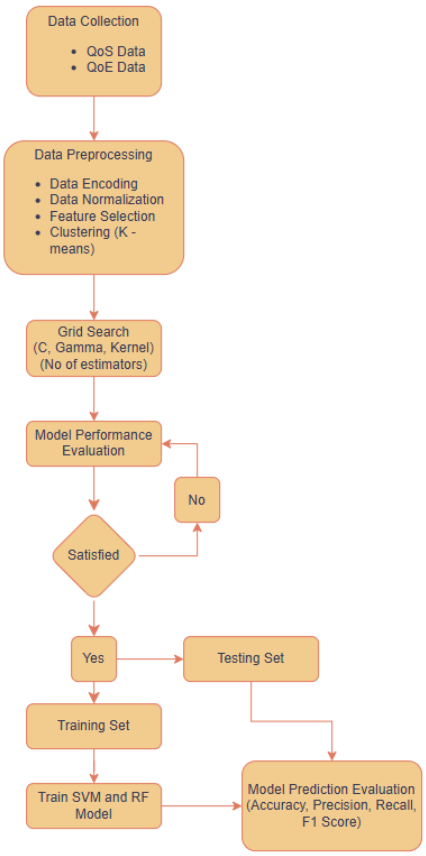

Figure 4.2: SVM and RF system model for predicting QoE

4.2 Data Collection

Technical and subjective data from both the ISP, ethio telecom, and the vISPs like Websprix IT Solution PLC and Zergaw Cloud are collected. To understand the subjective factors that influence customers, we used a Google Form to collect their opinions. At the same time, we gathered data about the QoS from ethio telecom customers using the PRTG system. For Websprix, we used the Observium system to get their QoS data, and for ZERGAW Cloud, we used the Ubiquity network management system.

Due to data limitations, particularly with vISPs like Zergaw, where we were unable

Service Provider Name	Sample Size
ethio telecom	313
Websprix IT Solution PLC	297
Zergaw Cloud	50

Table 4.1: Service provider and sample size

to obtain more than 50 data samples, this leads to a significant challenge. To address this, we carefully analyzed the available data and examined its distribution. Based on that understanding, we generated up to 1,000 synthetic data samples to improve the accuracy and robustness of our models.

4.3 Data Preprocessing

The data collected undergoes a preprocessing stage to get the data ready for the machine learning models. We first cleaned it up and organized it. Given that the subjective data collected entails categorical measurements, we converted those measurements into numbers so the models could understand them. It is like giving each category a special ID, and since the data had some values that were on very different scales, which could have unfairly influenced the models. To fix this, we standardized the data, essentially rescaling it to have a mean of zero and a standard deviation of one. This ensures that all the data contributes equally to the learning process and prevents any single value from dominating the results.

Then, the data were closely observed, and the hidden state and the observed sequence were defined. After that, we used clustering for dimensionality reduction. Clustering is

used to extract features by grouping similar data points. We used the K-means clustering algorithm, which is a popular, simple, and efficient algorithm for grouping data points into clusters and the silhouette coefficient method, which is a metric that measures how well each data point fits into its assigned cluster, to create a new feature or column that represents the observation sequence.

4.3.1 Feature Selection

Both technical and subjective datasets collected from ethio telecom and the vISPs have multiple parameters that needs feature selection. Therefore, we used Principal Component Analysis (PCA) for dimensionality reduction.

Principal Component Analysis (PCA) is a statistical technique for dimensionality reduction. It transforms a dataset with many variables to a dataset with fewer variables, called principal components, but preserves as much variation from the original data as possible.

In order to determine feature importance via PCA, we usually follow the following steps:

- Data Standardization: Standardize the data by taking away the mean and dividing by the standard deviation for each feature. This ensures that all features have equal weight in the analysis.

- PCA Model Fitting: Fit the PCA model to the standardized data. This generates a set of principal components, which are linear combinations of the original features, ordered by the amount of variance they explain.

- Calculate Loading Scores: Determine the loading scores for each feature on each principal component. Loading scores represent the contribution of each original feature to a particular principal component.

- Feature Importance Ranking: Rank features based on the absolute values of their loading scores on the most significant principal components. Features with high absolute loading scores on these components are considered more important.

Here are the feature selection based on loading score values for both ethio telecom and vISPs.

	PC1	PC2	PC3	PC4	PC5	PC6
average Internet service satisfaction	0.800479	-0.209627	0.006834	-0.045080	-0.044597	0.561460
Speed of Service Maintenace	0.101290	0.525938	0.514321	0.542809	-0.394535	0.048166
Quality of service maintenace	-0.094636	0.636397	-0.371744	-0.515730	-0.405512	0.146456
Time taken to reach call center agents	0.232215	0.560236	-0.404615	0.330372	0.602363	0.022970
Support given by call center agents	0.330713	0.284293	0.621805	-0.525224	0.352087	-0.165395
Overall satisfaction rate	0.761227	-0.064840	-0.268551	0.038463	-0.287737	-0.513765

(a) PCA for ET technical data

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Subscribe BW	0.954375	0.101218	0.015976	-0.038126	-0.022252	0.016275	0.283674
UP Time	0.048243	-0.044268	0.899783	0.435106	0.004759	0.027236	-0.006148
Average Latency/msec	0.023652	0.215772	-0.424293	0.880146	0.004750	-0.045794	0.009923
Average Packet loss	-0.114409	0.919951	0.014651	-0.051457	-0.066393	0.370173	-0.017844
Average Jitter	-0.181128	0.893319	0.132773	-0.129718	0.068121	-0.365896	0.014629
Average DL Speed/kbps	0.906116	0.094354	-0.000451	-0.017905	-0.374685	-0.086595	-0.160807
Average UL Speed/kbps	0.899350	0.091168	-0.025004	-0.020660	0.406010	0.043118	-0.138269

(b) PCA for ET technical data

Figure 4.3: Feature selection for ethio telecom

	PC1	PC2	PC3	PC4	PC5	PC6
average Internet service satisfaction	0.728583	-0.265309	-0.088733	-0.395876	0.483025	-0.071485
Speed of Service Maintenace	0.724369	-0.407373	-0.007927	-0.274002	-0.477927	-0.100099
Quality of service maintenace	0.575279	-0.201248	-0.382008	0.689280	0.057500	-0.091914
Time taken to reach call center agents	0.493232	0.061517	0.828581	0.249838	0.059151	-0.068634
Support given by call center agents	0.462902	0.847691	-0.172932	-0.115097	-0.059147	-0.157252
Overall satisfaction rate	0.932451	0.194565	-0.041267	0.021910	-0.043544	0.304694

(a) PCA for Websprix technical data

	PC1	PC2	PC3
Subscribed BW	0.894596	0.451591	-0.000753
Upload Speed	0.978347	-0.203514	0.075209
Download Speed	0.977088	-0.209689	-0.074616

(b) PCA for Websprix technical data

Figure 4.4: Feature selection for Websprix

According to the values of loading scores, features with the highest loading scores are considered as important feature. Figure 4.3a and 4.3b shows that average internet satisfaction, Overall satisfaction rate, Subscribe BW, Average DL Speed/kbps, and Average UL Speed/kbps have the largest loading scores and are selected for our analysis.

	PC1	PC2	PC3	PC4	PC5	PC6
average Internet service satisfaction	0.782809	-0.503886	-0.272943	0.045153	0.102376	0.269455
Speed of Service Maintenace	0.770871	-0.154486	0.610876	0.016140	-0.165812	0.085354
Quality of service maintenace	0.885212	-0.042041	0.084920	-0.367572	0.269126	-0.161858
Time taken to reach call center agents	0.705643	0.581908	0.048582	0.385914	0.193777	0.030445
Support given by call center agents	0.734518	0.537844	-0.245746	-0.269032	-0.238171	0.089603
Overall satisfaction rate	0.859095	-0.296737	-0.216732	0.236158	-0.177340	-0.256955

(a) PCA for Zergaw technical data

	PC1	PC2	PC3
Subscribed Speed	1.012236	-0.007147	0.040543
Average Upload Speed	1.011413	-0.052679	-0.024157
Average Download Speed	1.011170	0.059846	-0.016423

(b) PCA for Zergaw technical data

Figure 4.5: Feature selection for Zergaw

Figure 4.4a and 4.4b shows that average internet satisfaction, Speed of service maintenance, Overall satisfaction rate, Subscribed BW, Upload Speed, and Download Speed have the largest loading scores and are selected for Websprix analysis.

Figure 4.5a and 4.5b shows that average internet satisfaction, Speed of service maintenance, Quality of service maintenance, Time taken to reach call center agents, Support given by call center agents, Overall satisfaction rate, Subscribed speed, Average Upload Speed and Average Download Speed have largest loading scores and selected for Zergaw analysis.

4.3.2 Hidden State Formation

After feature selection, the next step is to define the hidden state, and to do that, we need to consider some rules of HMM.

1. Defining the states, the hidden state which is not observable to us, and the observable state that can be directly measured or observed by us.

2. There must be an emission from the hidden state to the observable state.

Here is the differentiation of the two parameters based on the above concepts.

To solve this dilemma, we considered the HMM as a simple mapping method and implemented both conditions, which is defining both QoS and QoE as hidden states vice

	Pros	**Cons**
QoS (hidden state) and QoE (observable state)	We can see the emission from the hidden state to the observable clearly	Physically the measured value/ easily observable is the QoS KPI
QoS (observable state) and QoE (hidden state)	QoS KPI is easily measurable from the provider side	Couldn't see the emission between the states

Table 4.2: QoS and QoE as hidden states based on HMM concept

versa and see the results of the algorithm. After we saw the results, we decided to go with QoE as a hidden state because we obtained higher accuracy with that.

After defining hidden states, the next step is to cluster. As we can see in the above feature selection section we have multiple parameters from both technical and subjective datasets. Therefore, we used K-means clustering for the hidden state formation.

We used the subjective dataset as a hidden state and clustered it and we got four classes as four hidden states based on the observation symbol we got from the K-means clustering.

User Satisfaction Level	Value
Very satisified	3
Satisified	2
Neutral	1
Dissatisfied	0

Table 4.3: Hidden states of the HMM

Here are the definitions of four hidden states in Table 4.3:

- *Very satisfied:* represents the highest level of user experience. When the HMM is in this state, it means that, based on the observed QoS, the user is experiencing the

service with virtually no issues. Performance is excellent, exceeding expectations.

- *Satisfied:* the user experience is still positive, but perhaps with some minor, tolerable imperfections. The service is performing well and meeting expectations.

- *Neutral:* indicates a borderline user experience. The user isn't particularly happy or unhappy with the service. Performance is adequate but may have some noticeable issues.

- *Dissatisfied:* represents a negative user experience. The user is experiencing significant problems with the service, leading to frustration and annoyance. Performance is poor and failing to meet expectations.

Once we had defined the hidden states (like the different levels of user satisfaction), we estimated the probabilities of transitions between those states using historical data. For example, we can count how many times a user satisfaction changed from the 'Satisfied' state to the 'Neutral' state, or from the 'Very satisfied' state to the 'Dissatisfied' state, and so on.

However, in this thesis, we used a Python tool called 'mchmm.' This tool included the four HMM algorithms that provided solutions to the HMM problems we discussed earlier. We used the package to implement Markov Chains and HMMs, and it also estimated all the necessary HMM parameters. Plus, it showed us a visual representation of how the user satisfaction states changed over time, along with the probabilities of those changes. For example, we used historical data of Overall service satisfaction values to visualize the Markovian pattern of the hidden states that would represent the four user satisfaction sequences as shown in Figure 4.6.

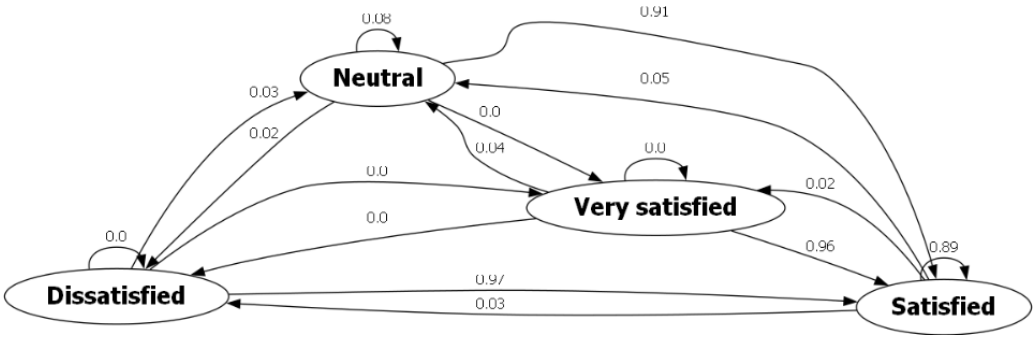

Figure 4.6: Transition probabilities

4.3.3 Observation Sequence Formation

In HMM-based prediction, the observed data must be transformed into a sequence of symbols (labels) before training and prediction because of the multiple parameters. To achieve this, we utilized a clustering-based approach. Specifically, we employed the K-means algorithm to group similar data points together, effectively creating new features that represent the observation sequence. To determine the optimal number of clusters for K-means, we employed the elbow method, which helps to identify the point of diminishing returns in the within-cluster sum of squares as the number of clusters increases.

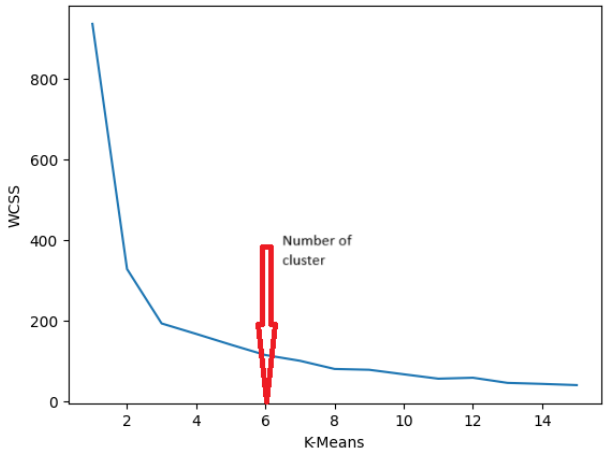

Figure 4.7: Optimal number of clusters using elbow method for ET

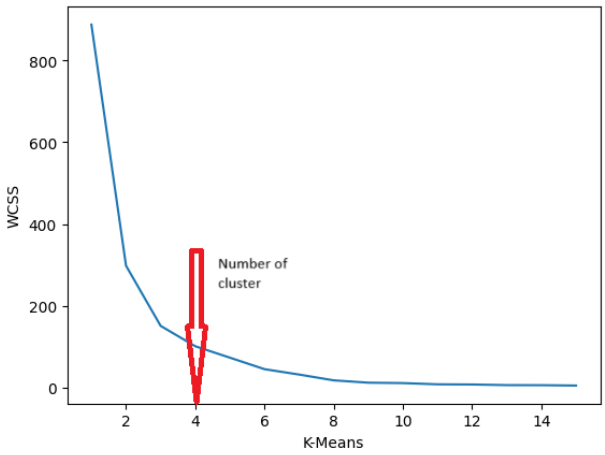

Figure 4.8: Optimal number of clusters using elbow method for Websprix

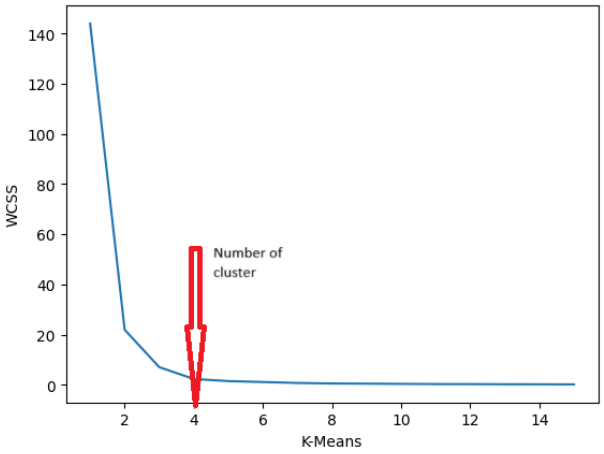

Figure 4.9: Optimal number of clusters using elbow method for Zergaw

We got the optimal number of clusters by using Elbow method and Figure 4.7 shows 6 clusters for ethio telecom, Figure 4.8 shows 4 clusters for Websprix IT Solution PLC and Figure 4.9 shows 4 clusters for Zergaw Cloud.

As shown in the figure, the elbow method indicates that six is the optimal number of clusters. Consequently, using K-means clustering, the normalized observation data were grouped into six unique observation sequences, as detailed in the table.

	Subscribe BW	Average DL Speed/kbps	Average UL Speed/kbps	Observation_Symbols
0	0.033043	0.087744	-0.100855	0
1	0.722540	0.817849	-0.070065	4
2	0.722540	0.435817	-0.016182	4
3	0.033043	0.444307	-0.308689	4
4	-0.656454	-0.319756	-0.108553	0
...	...	...	...	...

Figure 4.10: Formation of a unique number of observation symbols

The Observation_Symbols column in the Figure 4.10 identifies these six clusters, labeled from 0 to 5, which correspond to sequential, discrete observation symbols. A sample output of these sequential observations is represented as a NumPy array: array([[0],[4],[4],[4],[0],...). In this array, each number represents a symbol within the observation sequence.

Chapter 5: Results and Discussion

The observed results and their interpretation are the main topics of this chapter. It begins with a thorough explanation of model training and validation, covering the complexity of the initial probability distribution, transition probability matrix, emission probability matrix, and hyperparameter optimization procedure, and then performance is evaluated using important metrics like accuracy, precision, recall, and F1 score. Finally, hidden state prediction with a comparison of predicted results offers a thorough examination of the model's output.

5.1 Model Training and Validation

After data preprocessing and feature selection, we used the Baum-Welch algorithm which is an iterative algorithm that was used to train the HMM model by starting with initial estimates of the parameters, and then iteratively updating the estimates until they converge or estimate the three parameters of an HMM. The "thres" argument specifies the threshold for convergence of the iteration. This argument is the minimum change in the model's parameters (log-likelihood of the model) that is required before the algorithm terminates. We set the value of "thres" to be 0.00001. Convergence is assumed to occur if

the change in log-likelihood is less than the predefined threshold, set at 0.00001.

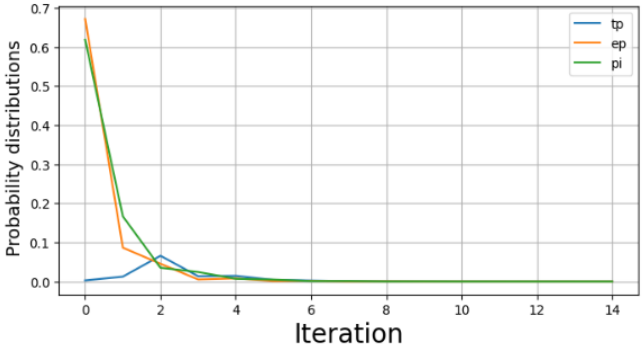

Figure 5.1: Baum Welch's model convergence

The final output of the Baum-Welch algorithm was the optimized HMM parameters, which were the transition probabilities, prior (or initial) probabilities, and emission probabilities. Those were crucial parameters used to predict the future state of user satisfaction and describe the HMM in terms of probability. Thereafter, the Viterbi algorithm was utilized to predict the most likely sequence of hidden states. The algorithm recursively calculated the probability of each hidden state at each time step based on the previous observations and hidden states. The most likely sequence of states was the one with the highest probability.

5.1.1 Transition Probability Matrix

The transition probability matrix is a square matrix that describes the probability of transitioning from one hidden state to another.

Matrix Description: In a transition probability matrix, The current hidden state is represented by the rows, and the next potential hidden state is indicated by the columns. Each element within the matrix denotes the probability of transitioning from state i (the row) to state j (the column) in a single time step.

Properties: Each column of the transition matrix must sum to 1, which ensures

that the system will transition to one of the possible next states with absolute certainty. In some cases, the matrix can be doubly stochastic, meaning that both the rows and columns sum to 1. While the order of states in the rows and columns can be assigned arbitrarily. It is absolutely crucial that this order be preserved throughout the analysis.

For example, in Figure 5.2a, the value 0.75972 on the second row and first column in the matrix shows us that there is a 75.972% chance that the user satisfaction is in the "Satisfied" state will make a transition to the "Very satisfied" state in the next t + 2 time step. This means that the system is more likely to transit to the "Very satisfied" state than staying in the "Satisfied" state.

	Very satisfied	Satisfied	Neutral	Dissatisfied
Very satisfied	0.845900	0.093335	0.002906	0.057859
Satisfied	0.759719	0.150442	0.014934	0.074905
Neutral	0.556428	0.364253	0.007640	0.071679
Dissatisfied	0.609496	0.289508	0.012409	0.088588

(a) Transition probability matrix for ethio telecom

	Very satisfied	Satisfied	Neutral	Dissatisfied
Very satisfied	0.0	0.36803	0.00751	0.62446
Satisfied	0.0	0.26486	0.02525	0.70989
Neutral	0.0	0.06690	0.03103	0.90207
Dissatisfied	0.0	0.14080	0.02650	0.83270

(b) Transition probability matrix for Websprix

	Very satisfied	Satisfied	Neutral	Dissatisfied
Very satisfied	0.97669	0.01356	0.00429	0.00546
Satisfied	0.84037	0.05054	0.10888	0.00021
Neutral	0.92571	0.01485	0.05555	0.00389
Dissatisfied	0.85500	0.04669	0.08720	0.01111

(c) Transition probability matrix for Zergaw

Figure 5.2: Transition probability matrix for ethio telecom, Websprix and Zergaw

For instance, the values in the matrix Figure 5.3c show a high probability of transitioning (0.97669 and 0.85500) "Very satisfied" to "Very satisfied" and "Dissatisfied" to "Very satisfied" respectively. In general, the numbers in the first column of the matrix show us that the system is more likely to stay in a "Very satisfied" state. This is somewhat reassuring, but not perfect. It means that the system tends to be quite stable and functional

for extended periods of time, most of it in this "Very satisfied" user satisfaction state.

5.1.2 Emission Probability Matrix

The emission probability is the probability of emitting a particular observation given a hidden state. The emission probability matrix shows us the likelihood of observing a particular value given a specific state. For example, the illustration in Figure 5.3a shows the emission probability of the four-state user satisfaction level with six clusters of cells (0, 1, 2, 3, 4, 5), which are unique observation symbols.

	0	1	2	3	4	5
Very satisfied	0.32959	0.0352	0.31681	0.26400	0.016	0.0384
Satisfied	0.32959	0.0352	0.31681	0.26400	0.016	0.0384
Neutral	0.32959	0.0352	0.31681	0.26400	0.016	0.0384
Dissatisfied	0.32962	0.0352	0.31678	0.26401	0.016	0.0384

(a) Emission probability matrix for ethio telecom

	0	1	2	3
Very satisfied	0.43319	0.01687	0.43185	0.11809
Satisfied	0.43345	0.01686	0.43166	0.11803
Neutral	0.43345	0.01686	0.43166	0.11803
Dissatisfied	0.43345	0.01686	0.43166	0.11803

(b) Emission probability matrix for Websprix

	0	1	2	3
Very satisfied	0.57734	0.08246	0.05158	0.28862
Satisfied	0.57730	0.08249	0.05152	0.28869
Neutral	0.57733	0.08247	0.05156	0.28864
Dissatisfied	0.57731	0.08248	0.05152	0.28869

(c) Emission probability matrix for Zergaw

Figure 5.3: Emission probability matrix for ethio telecom, Websprix and Zergaw

5.1.3 The Initial Probability Distribution

The initial probability (π) is the probability of starting in a particular state. For example, the initial probability matrix in Figure 5.4a shows us the probability that the HMM will start in each of the four hidden state. The value of 0.24933 for the "Very

satisfied" state means that there is a 24.93% chance that the system will start in the "Very satisfied" state. Similarly, the value of 0.252783 for the "Dissatisfied" state means that there is a 25.27% chance that the system will start in the "Dissatisfied" state, and so on.

	Very satisfied	Satisfied	Neutral	Dissatisfied
0	0.24933	0.248841	0.249047	0.252783

(a) Initial probability distribution for ethio telecom

	Very satisfied	Satisfied	Neutral	Dissatisfied
0	0.244431	0.251708	0.251977	0.251884

(b) Initial probability distribution for Websprix

	Very satisfied	Satisfied	Neutral	Dissatisfied
0	0.25134	0.249063	0.250455	0.249142

(c) Initial probability distribution for Zergaw

Figure 5.4: Initial probability distribution for ethio telecom, Websprix and Zergaw

Support Vector Machine and Random Forest models are trained using k-fold cross-validation with the selected influential factors and their optimal hyperparameter values. K-fold cross-validation is a resampling technique that helps estimate how well a model will perform on unseen data. Instead of splitting the data into a single training and testing set, the data is divided into 'k' distinct subsets (folds). The model is then trained on 'k-1' folds and evaluated on the remaining fold. This process is repeated 'k' times, with each fold serving as the validation set exactly once. The final performance metric is the average of the 'k' scores obtained from each fold. In this thesis, we used three k-folds to train the models.

For the Random Forest model, we used a technique called Bootstrap Aggregating (or 'bagging') during training. After training both the Support Vector Machine and Random Forest models, we wanted to see how well they performed. So, we used cross-validation, which gives us an average score of each model's performance.

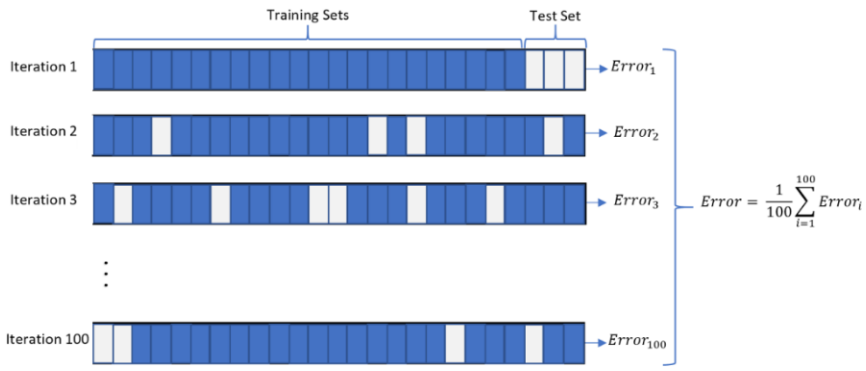

Figure 5.5: K - fold cross validation model training [38]

5.1.4 Hyperparameter Optimization

We used Grid search for SVM and RF to explore the most promising hyperparameter ranges. Grid search is a hyperparameter tuning method in machine learning that systematically tries out every possible combination of values from a predefined set of hyperparameters. It creates a 'grid' of these combinations, evaluates the model's performance for each one, and then selects the combination that produces the best results. While this method is thorough and can find the optimal settings, it can be computationally expensive, especially when dealing with many hyperparameters or large datasets.

Support Vector Machine Hyperparameters Initialization:

- C: [0.1, 1, 10, 100, 1000]
- Gamma': [1, 0.1, 0.01, 0.001, 0.0001, 'scale']
- Kernel': ['linear', 'poly', 'rbf', 'sigmoid']

Random Forest Hyperparameter Initialization:

- n_estimators: [20, 50, 100]

- used the default value for other hyperparameters (max depth, min samples split, and others), considering the high computational cost.

Service Providers	C	Gamma	Kernel	n_estimators
ethio telecom	0.1	1	Linear	100
Websprix	100	1	Linear	100
Zergaw	100	0.001	rbf	100

Table 5.1: Grid search results for different service providers

5.2 Model Evaluation

To really test how good these models were at predicting, we used a separate test dataset. From the entire dataset, 80% is utilized for training the model, while the remaining 20% is allocated for testing purposes. We then looked at accuracy, precision, recall, and F1 score.

5.2.1 Accuracy

Accuracy is a fundamental performance metric in machine learning, particularly for classification models. It quantifies the overall correctness of a model by measuring the proportion of correctly predicted instances out of the total number of observations [37].

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{5.1}$$

Where,

True Positive (TP): When the model correctly predicts a positive class for an instance that actually belongs to the positive class.

True Negative (TN): When the model correctly predicts a negative class for an instance that actually belongs to the negative class.

False Positive (FP): When the model incorrectly predicts a positive class for an instance that actually belongs to the negative class.

False Negative (FN): When the model incorrectly predicts a negative class for an instance that actually belongs to the positive class.

In the above computation it doesn't consider weighted measures meaning it computes by hard decision. Therefore, we used a distance matrix for the weighted average method. The distance matrix naturally captures the ordinal nature of the satisfaction levels. "Very Satisfied" is closer to "Satisfied" than to "Dissatisfied" and so on. And the formula is:

$$WeightedAccuracy = 1 - \left[\frac{\sum_{i=1}^{N} d(Y_{true}(i), Y_{pred}(i))}{(N) * max(d)} \right] \tag{5.2}$$

Where,

N: The length of the sequence

d: The distance matrix

Y_{true}: The true state

Y_{pred}: The predicted state

5.2.2 Precision

Precision is a performance metric that measures the proportion of true positive predictions among all instances predicted as positive by the model. It essentially indicates the reliability of the positive predictions made by the model [37].

$$Precision = \frac{(TP)}{(TP + FP)} \tag{5.3}$$

5.2.3 Recall

Recall, also known as sensitivity, is a performance metric that measures the proportion of true positive predictions among all actual positive instances. It indicates the model's

ability to identify all relevant positive instances, essentially showing how well it captures all true positives [37].

$$Recall = \frac{(TP)}{(TP + FN)} \tag{5.4}$$

5.2.4 F1 Score

The F1-score is the harmonic mean of precision and recall. It provides a single, balanced metric to evaluate a model's performance when both precision and recall are equally important. This is particularly valuable in scenarios with imbalanced datasets where both false positives and false negatives have significant consequences [37].

$$F1Score = \frac{(2 * Precision * Recall)}{(Precision + Recall)} \tag{5.5}$$

5.3 Hidden State Prediction

With HMMs, a key task is figuring out the most likely sequence of hidden states – think of it as uncovering the underlying story behind what we actually observe. We use the Viterbi algorithm for this, which is a clever technique that works step-by-step to find the most probable sequence of these hidden states, given the sequence of observations we've recorded.

The Viterbi algorithm works by carefully calculating, at each point in time, the likelihood of each hidden state, considering both what we've seen so far and the possible previous states. It's like piecing together clues to find the most coherent explanation. The sequence of hidden states with the highest overall probability is our best guess at what was

really happening.

Once we've trained our HMM, we can use it to predict these hidden state sequences, which, in our case, represent the 'satisfaction' of a user over time. To test this, we create a sample sequence of observable events and the corresponding true hidden states using our trained HMM. We set a specific length for this observation sequence.

It's important to understand that the sequence of observed events is determined by the HMM's emission and transition probabilities. These probabilities, which are learned during the training process (represented by the model parameters λ, consisting of A, B, and π), dictate how likely we are to see a particular observation given a hidden state and how likely the hidden states are to change over time.

In our implementation, we use a function called HiddenMarkovChain_Uncover(A, B, π) to set up the HMM. Then, the function observed_sequence, latent_sequence = hmm.run(length = 4) simulates the HMM for a specified observation length (in this case, 4). The observed_sequence variable holds the generated sequence of observations, and latent_sequence holds the actual hidden states that produced those observations. Both sequences have a length of 1 + N, where N (which is 4 here) is the observation length we've chosen for our evaluation.

Finally, the ypredicted = hmm.uncover (observed_sequence) function takes the observed sequence as input and uses our trained HMM (hmm) to predict the most likely sequence of hidden states that generated those observations. Under the hood, this function uses the Viterbi algorithm to find the most probable sequence.

The table shows that the predicted hidden state sequence matches the true hidden state sequence for the given observation sequence. This indicates accurate hidden state

Initial distribution	Length of Observation	Observation Sequence	True Hidden State Sequence	Predicted Hidden State Sequence	Performance of the Model			
Very satisfied Satisfied Neutral Dissatisfied					Accuracy	Precision	Recall	F-1
[1 0 0 0]	5	[3, 4, 3, 3, 0]	Very satisfied, Dissatisfied, Dissatisfied, Dissatisfied, Dissatisfied	Very satisfied, Dissatisfied, Dissatisfied, Dissatisfied, Dissatisfied	100	100	100	100
[0 1 0 0]	5	[0, 0, 3, 3, 3]	Satisfied, Very satisfied, Dissatisfied, Very satisfied, Very satisfied	Satisfied, Very satisfied, Dissatisfied, Very satisfied, Very satisfied	99.8	100	100	100
[0 0 1 0]	5	[0, 4, 0, 5, 2]	Neutral, Dissatisfied, Neutral, Satisfied, Satisfied	Neutral, Dissatisfied, Neutral, Satisfied, Satisfied	99.6	100	100	100
[0 0 0 1]	5	[3, 4, 3, 0, 0]	Dissatisfied, Dissatisfied, Neutral, Satisfied, Satisfied	Dissatisfied, Dissatisfied, Neutral, Satisfied, Satisfied	100	100	100	100
[0.252 0.25 0.247 0.248	100	Observation Sequence = ['2', '4', '3', '3', '0', '0', '0', '3', '3', '3', '4', '4', '0', '3', '1', '3', '3', '3', '3', '4', '4', '0', '0', '0', '3', '3', '0', '4', '4', '4', '4', '0', '3', '3', '0', '4', '3', '3', '0', '4', '4', '4', '3', '0', '0', '3', '4', '3', '0', '0', '0', '0', '3', '3', '3', '4', '0', '2', '4', '4', '3', '0', '0', '4', '0', '3', '3', '0', '4', '0', '0', '0', '3', '3', '4', '3', '3', '4', '3', '3', '0', '4', '4', '4', '1', '3', '4', '3', '0', '0', '3', '0', '4', '0', '3', '3', '3', '4', '5', '0']	True Hidden = ['Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Dissatisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Dissatisfied', 'Very satisfied', 'Dissatisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Dissatisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied']	Predicted Hidden State Sequence: ['Very satisfied', 'Very satisfied']	94	62.4	79	69.7
	200	Observation Sequence = ['4', '4', '4', '3', '3', '5', '0', '0', '3', '0', '3', '4', '4', '4', '5', '0', '0', '4', '3', '0', '4', '3', '0', '3', '3', '0', '4', '4', '0', '4', '4', '3', '4', '0', '0', '3', '0', '3', '4', '0', '4', '0', '3', '3', '4', '4', '4', '0', '4', '0', '0', '0', '4', '0', '0', '3', '3', '4', '4', '0', '0', '4', '0', '0', '0', '3', '3', '4', '3', '4', '3', '3', '0', '3', '4', '3', '0', '4', '0', '0', '3', '3', '0', '0', '0', '3', '3', '3', '4', '4', '3', '4', '0', '3', '0', '0', '5', '3', '0', '4', '0', '3', '4', '3', '4', '0', '3', '0', '4', '0', '3', '3', '3', '0', '1', '0', '3', '0', '2', '0', '3', '4', '3', '0', '3', '3', '5', '3', '0', '4', '5', '0', '3', '4', '4', '4', '0', '3', '4', '4', '0', '0', '3', '0', '2', '0', '3', '4', '4', '0', '4', '5', '3', '0', '0', '4', '2', '4', '0', '3', '0', '0', '2', '0', '5', '3', '4', '3', '0', '0', '0', '4', '3', '2', '1', '3', '0', '3', '3', '0', '3', '3', '0', '5', '3', '0', '4', '3', '2', '0', '4', '3', '4', '0', '0', '3', '3', '3', '4', '5']	True Hidden = ['Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Dissatisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Dissatisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Dissatisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Dissatisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Dissatisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Dissatisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Neutral', 'Satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Very satisfied', 'Satisfied', 'Very satisfied']	Predicted Hidden State Sequence: ['Very satisfied', 'Very satisfied']	93.8	97.75	97	98.5

Figure 5.6: QoE prediction output and model performance of HMM

prediction by the HMM, suggesting effective training and suitability for reliable hidden sequence prediction. Here is a more detailed interpretation of the output in Table. 5.6:

- Initial Probability: This is the probability that the HMM is in the very satisfied state at the beginning of the sequence. In this case, the Initial Probability: [1 0 0 0] suggests prediction of user satisfaction using HMM, where the initial state of the HMM was in the normal state with a probability of 1. This indicates the assumption of the user is very satisfied. In addition, we used the initial probability generated

by the Baum-Welch algorithm to predict the hidden state sequence, where the initial probability is a vector: [0.252, 0.25, 0.247, 0.248].

- Length of observation: is the number of observation sequences, and in this case, we choose to be 5, but in the next cases, we can see up to 200 sequences.

- Observation sequence: is the sequence of cluster numbers that was observed. In this case, the observation sequence is [3, 4, 3, 3, 0]. Each number in the observation sequence represents a group of QoS of the service providers.

- True hidden state sequence: The true hidden state sequence represents the actual user satisfaction levels of the user that actually generated the observation sequence at each time step. In this case, the true hidden state sequence is: ['Very satisfied', 'Dissatisfied', 'Dissatisfied', 'Dissatisfied', 'Dissatisfied'].

- Predicted Hidden State Sequence: This is the sequence of hidden states that the HMM predicted from the generated observation sequence. In this case, the predicted hidden state sequence is also ['Very satisfied', 'Dissatisfied', 'Dissatisfied', 'Dissatisfied', 'Dissatisfied']. The HMM was able to accurately predict the hidden states, which means that it is likely that the user is dissatisfied.

5.4 Comparison of Predicted Results

We used weighted accuracy for HMM, and we created a confusion matrix table for SVM and RF, the main internet provider (ethio telecom) and the virtual providers (Websprix IT Solution and Zergaw Cloud).

Confusion matrix (or error matrix) is a visualization method for classifier algorithm results. More specifically, a confusion matrix is a table that breaks down the number of ground truth instances of a specific class against the number of predicted class instances. Confusion matrices are one of several evaluation metrics that are used to measure the performance of a classification model in machine learning [36].

Confusion Matrix of QoE Mapping Technique 1 for ethio telecom, Websprix, and Zergaw

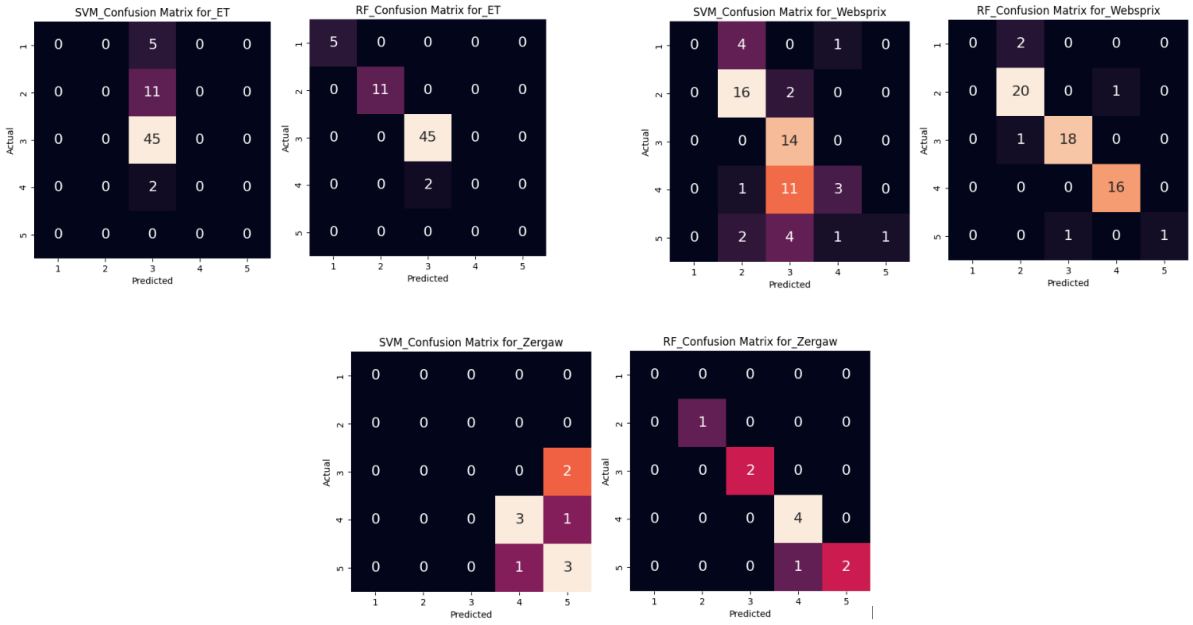

Figure 5.7: Confusion matrix for ethio telecom, Websprix, and Zergaw

Confusion Matrix of QoE Mapping Technique 2 for ethio telecom, Websprix, and Zergaw

Figure 5.8 presents confusion matrices for ethio telecom, Websprix, and Zergaw evaluated using SVM and RF classifiers. For ET, both models demonstrate strong performance, particularly in correctly classifying class 3 instances. The RF model slightly outperforms

SVM with more consistent predictions and fewer misclassifications. In the case of Websprix, RF achieves better predictive accuracy with minimal confusion between neighboring labels. SVM, on the other hand, shows notable misclassifications, indicating weaker decision boundaries. For Zergaw, the performance of both models is more limited, largely due to the small dataset size.

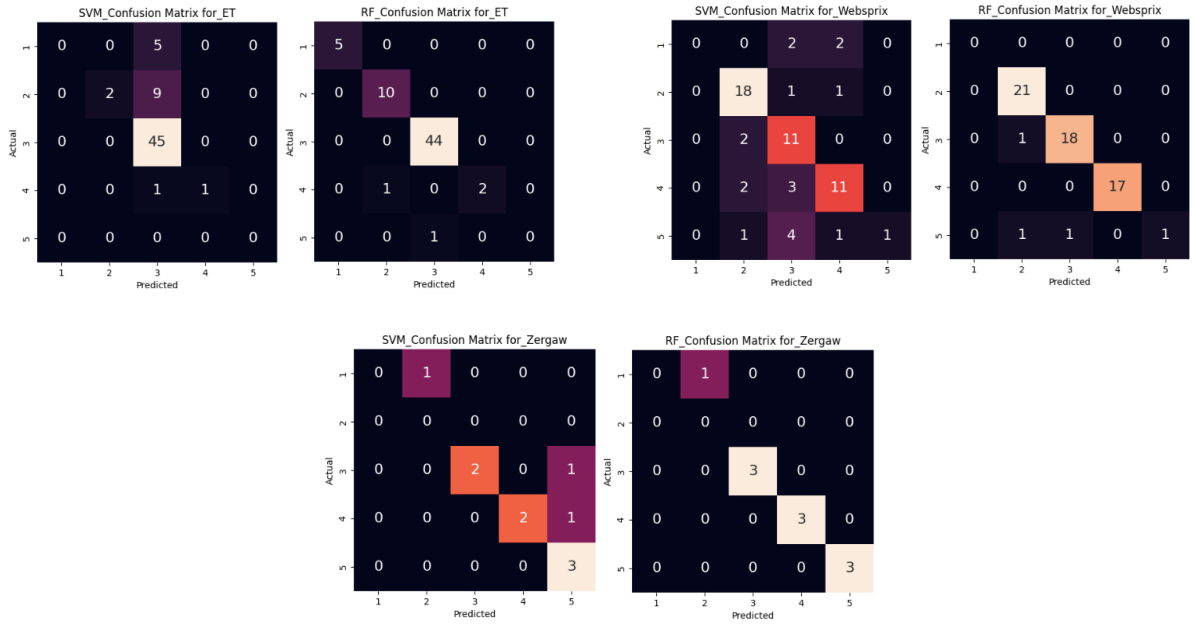

Figure 5.8: Confusion matrix for ethio telecom, Websprix, and Zergaw

Confusion Matrix of QoE Mapping Technique 3 for ethio telecom, Websprix, and Zergaw

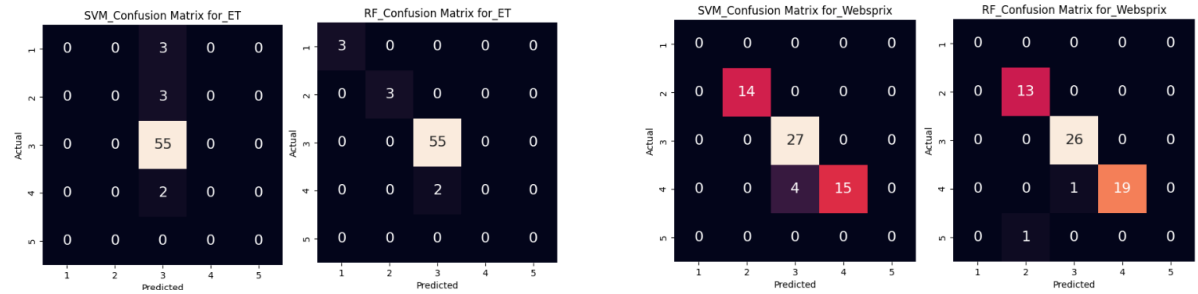

Figure 5.9: Confusion matrix for ethio telecom and Websprix

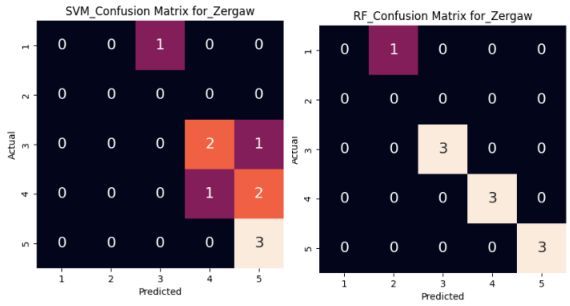

Figure 5.10: Confusion matrix for Zergaw

Finally, we compared the results from three models to figure out which one was the best. The graph presented in the following graphs illustrates the calculated values for accuracy, precision, recall, and the F1 score. These metrics were obtained from the confusion matrix and evaluated independently for the three QoE modeling approaches mentioned earlier. Hidden Markov Model, Support Vector Machine, and Random Forest models were analyzed using these performance indicators to offer a thorough evaluation of their effectiveness across the different modeling scenarios.

Model Prediction Evaluation for QoE Modeling Technique 1 (Mapping QoS (Technical Data) to MOS value)

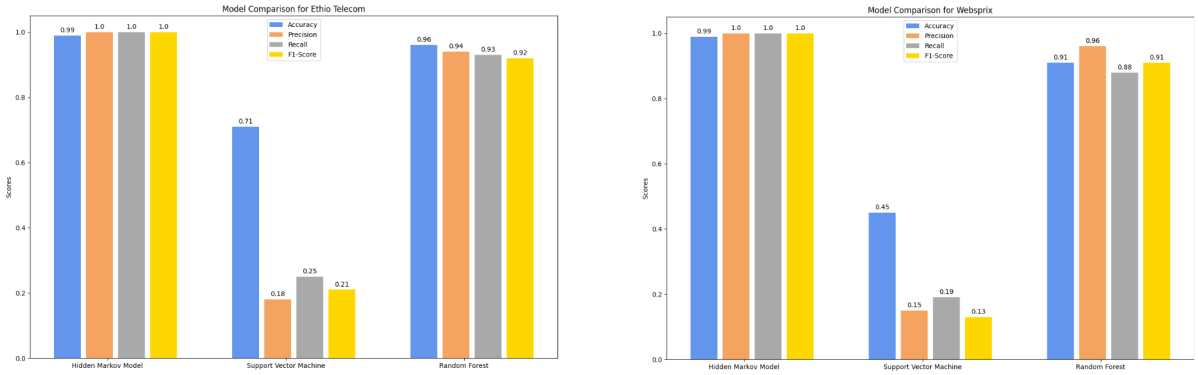

Figure 5.11: Model prediction evaluation for ethio telecom and Websprix

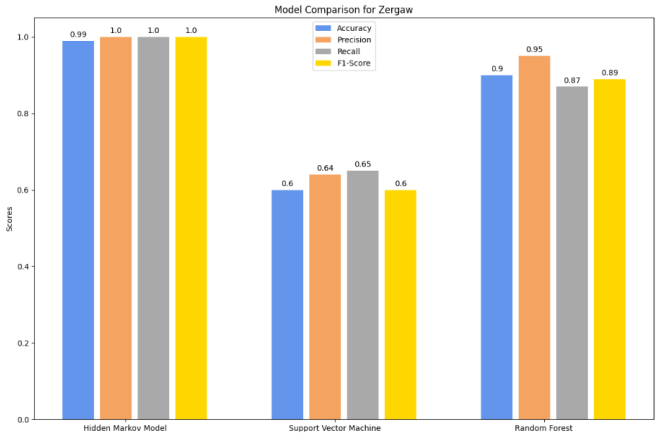

Figure 5.12: Model prediction evaluation for Zergaw

In the above results, HMM achieves near-perfect scores (around 100%) in all scenarios, demonstrating a strong ability to accurately model the relationship between QoS and QoE. Random Forest also performs very well, with accuracy scores 96%, 91%, and 90% for ET, Websprix, and Zergaw, respectively. In contrast, SVM consistently shows significantly lower performance with an accuracy score 71%, 45%, and 60% for ET, Websprix, and Zergaw, respectively.

Model Prediction Evaluation for QoE Modeling Technique 2 (Mapping QoS (Technical Data) plus QoE (Subjective Data) to MOS Value)

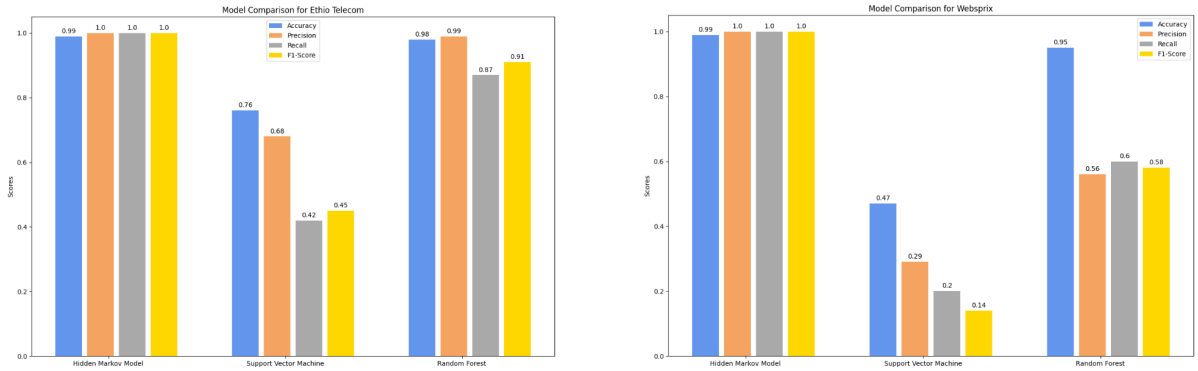

Figure 5.13: Model prediction evaluation for ethio telecom and Websprix

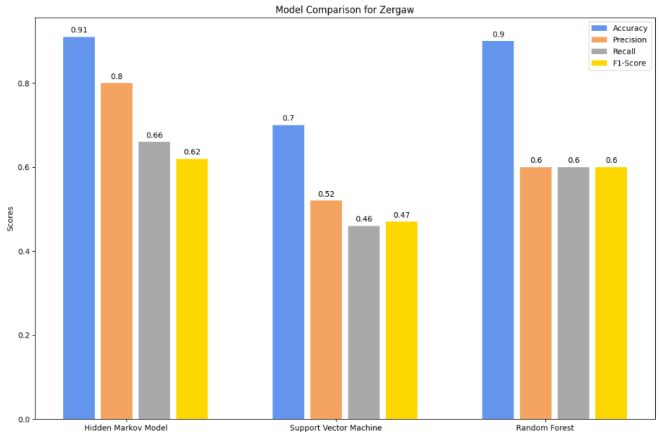

Figure 5.14: Model prediction evaluation for Zergaw

For ethio telecom, the HMM achieves an accuracy of 99%, with precision, recall, and F1 score all reaching 100%. The RF model scores 98% accuracy, 99% precision, 87% recall, and 91% F1 score. In contrast, the SVM model yields 76% accuracy, 68% precision, 42% recall, and 45% F1 score. Similarly, for Websprix and Zergaw, the HMM model shows strong performance, while the RF model performs slightly worse compared to its results on ethio telecom.

Model Prediction Evaluation for QoE Modeling Technique 3 (Mapping QoS (Technical Data) to QoE (Subjective Data))

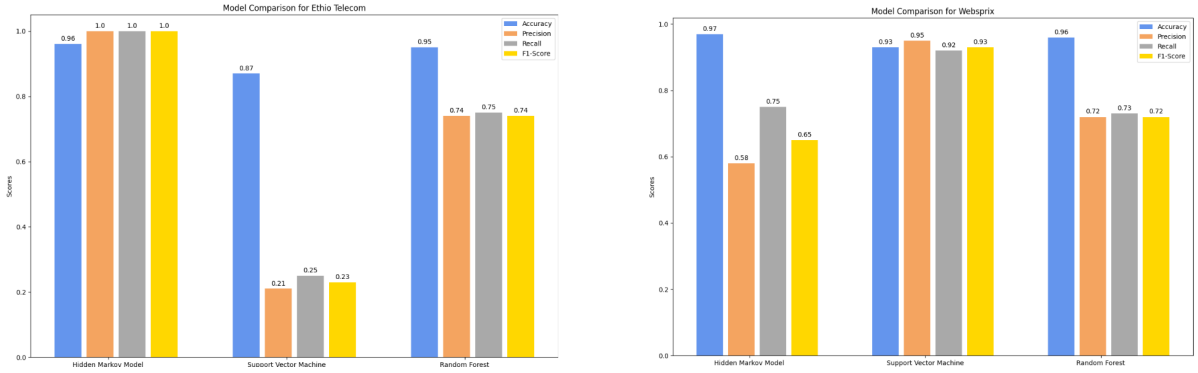

Figure 5.15: Model prediction evaluation for ethio telecom and Websprix

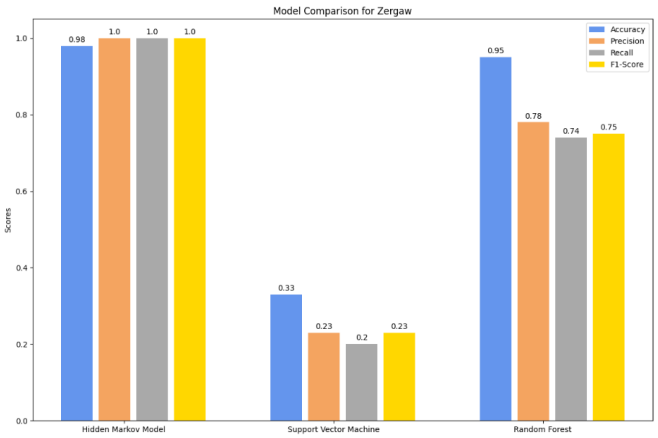

Figure 5.16: Model prediction evaluation for Zergaw

Across ethio telecom, Websprix, and Zergaw datasets, the HMM achieves near-perfect to perfect scores across Accuracy, Precision, Recall, and F1-Score, indicating a strong capability in capturing the underlying patterns. Random Forest also demonstrates robust performance in all three scenarios, consistently yielding high scores across the evaluation metrics, often closely approaching the HMM's results. In stark contrast, the SVM consistently underperforms across all three scenarios, exhibiting lower Accuracy and notably weaker Precision, Recall, and F1-Scores compared to both HMM and Random Forest, suggesting its limited effectiveness in modeling the QoS-to-QoE relationship within these specific datasets.

Chapter 6: Conclusions and Future Work

6.1 Conclusions

This thesis has demonstrated the effectiveness of HMMs in predicting the QoE by capturing the temporal dynamics and latent states of user satisfaction within virtual ISP networks, and using the data from ethio telecom and two virtual ISPs (Websprix IT Solution PLC and ZERGAW Cloud). In order to make sure that our machine learning algorithm could deal with this data efficiently and objectively, we carried out some initial processing. We used a label encoder to convert any text-based category into numeric values that the algorithm could handle, and normalized all measurements. This step was crucial because it prevents the model from being biased by differences in how various things were measured. Then, to train an HMM model, the Viterbi and Baum-Welch algorithms are used in the HMM framework, whereas for SVM and RF, we used the Grid Search technique to determine the best hyperparameter values and employed K-fold cross-validation to ensure the model learned reliably from the selected factors. We used three QoE modelling assessments for this thesis, and the output of the trained model consists of the estimated HMM parameters, which can be used to understand the user satisfaction. The evaluation of model performance for HMM, RF, and SVM with scores 99%, 96%,

and 71%, 99%, 91%, and 45%, and 99%, 90%, and 60% for ethio telecom, Websprix, and Zergaw, respectively.

In QoS to QoE mapping across ethio telecom, Websprix, and Zergaw scenarios, HMM and RF perform better than SVM in all scenarios. This implies that while HMM and RF correctly and effectively capture the underlying patterns in the data, SVM is not as effective in correctly mapping QoS to QoE in these scenarios. The three models comparison results show that HMMs are a promising tool for predicting QoE because HMMs perform superbly in comprehending sequential data and how it changes over time, which is in line with the behavior of network data. In particular, HMMs work well because they model the temporal dependencies inherent in network behavior explicitly, and they are grounded in the Markovian process, which is a reasonable assumption for this kind of data.

6.2 Future Work

Based on the analysis in this thesis, we would like to suggest these areas for future research. The study could be extended with a larger and more diverse dataset, which includes data from more diverse user demographics, network conditions, and service types, to improve the generalizability of the models. Given that this thesis compared three machine learning models, another suggestion for future work is to use hybrid modeling approaches. Combining HMMs with deep learning, for example, could enhance predictive capability by first modeling the temporal structure probabilistically, then improving predictions with data-driven deep learning methods to take advantage of their respective strengths and address limitations inherent in individual models.

Appendix A: Comparision Results

The results below show the transition probability, Emission probability, and initial distribution probability of ethio telecom, Websprix and Zergaw using HMM model.

	Very satisfied	Satisfied	Neutral	Dissatisfied
Very satisfied	0.03802	0.03939	0.91626	0.00633
Satisfied	0.00895	0.00551	0.98314	0.00240
Neutral	0.00830	0.00290	0.98845	0.00035
Dissatisfied	0.07513	0.02868	0.89106	0.00513

	0	1	2	3	4	5
Very satisfied	0.29118	0.16641	0.29121	0.016	0.16159	0.07360
Satisfied	0.29123	0.16639	0.29118	0.016	0.16161	0.07359
Neutral	0.29123	0.16639	0.29117	0.016	0.16162	0.07359
Dissatisfied	0.29116	0.16642	0.29124	0.016	0.16158	0.07361

	Very satisfied	Satisfied	Neutral	Dissatisfied
0	0.248505	0.252195	0.252997	0.246302

Figure A.1: Transition probability, emission probability, and initial distribution probability of ethio telecom

	0	1	2	3
Very satisfied	1.770983	0.039756	0.071561	0.103362
Satisfied	1.790952	0.040200	0.072359	0.104521
Neutral	1.786492	0.040101	0.072181	0.104263
Dissatisfied	1.793575	0.040258	0.072464	0.104672

	0	1	2	3
Very satisfied	0.89189	0.02002	0.03604	0.05205
Satisfied	0.89189	0.02002	0.03603	0.05205
Neutral	0.89189	0.02002	0.03604	0.05205
Dissatisfied	0.89190	0.02002	0.03603	0.05205

	Very satisfied	Satisfied	Neutral	Dissatisfied
0	0.24657	0.251201	0.250234	0.251996

Figure A.2: Transition probability, emission probability, and initial distribution probability of Websprix

	Very satisfied	Satisfied	Neutral	Dissatisfied
Very satisfied	0.43246	0.02251	0.54397	0.00106
Satisfied	0.62841	0.02677	0.32901	0.01581
Neutral	0.97450	0.01513	0.00074	0.00963
Dissatisfied	0.89764	0.05437	0.03635	0.01165

	0	1	2	3
Very satisfied	0.11010	0.66369	0.10810	0.11811
Satisfied	0.11011	0.66366	0.10811	0.11812
Neutral	0.11010	0.66369	0.10810	0.11811
Dissatisfied	0.11013	0.66362	0.10812	0.11813

	Very satisfied	Satisfied	Neutral	Dissatisfied
0	0.252449	0.248761	0.252358	0.246432

Figure A.3: Transition probability, emission probability, and initial distribution probability of Zergaw

The figures below show the average performance of the models through three-fold cross-validation for both Support Vector Machine and Random Forest.

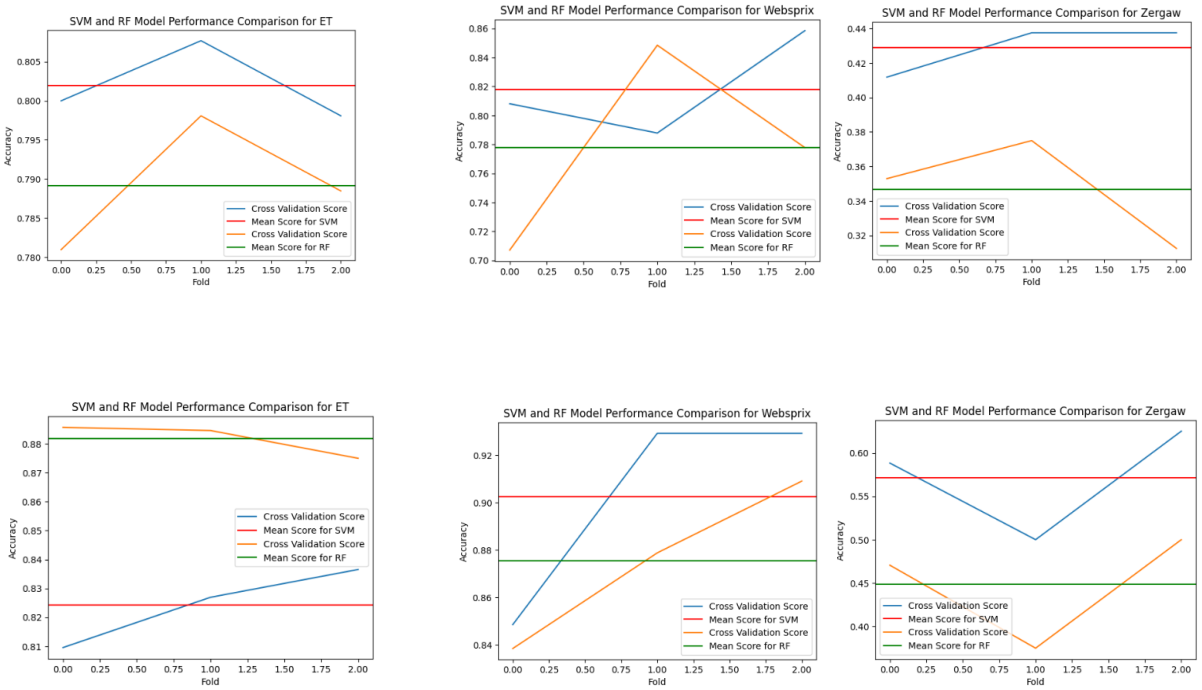

Figure A.5: k fold cross validation

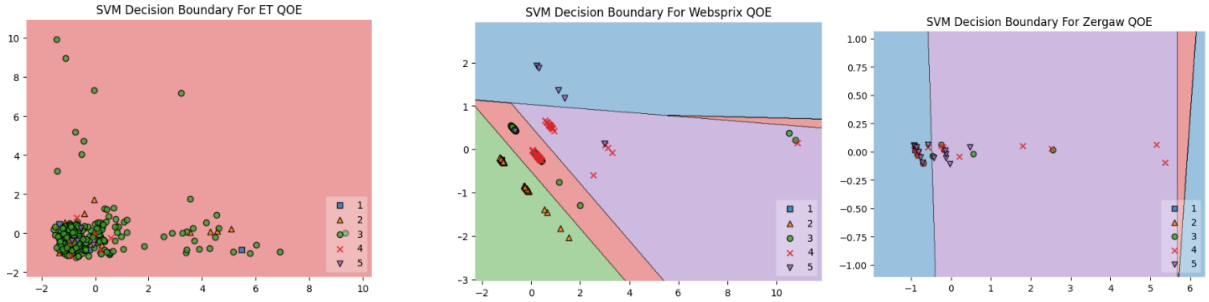

Figure A.6: Decision boundery for ethio telecom, Websprix and Zergaw

Bibliography

[1] M. Yang, S. Wang, R. N. Calheiros, and F. Yang, "Survey on QoE assessment approach for network service," *IEEE Access*, vol. 6, pp. 48374–48390, 2018.

[2] O. Markaki, D. C. & Nikitopoulos, D., "Enhancing Quality of Experience in Next-Generation Networks through Network Selection Mechanisms," in *Proc. IEEE 18th Int. Symp. on Personal, Indoor and Mobile Radio Communications*, 2007, pp. 1–5.

[3] K. Bouraqia, E. Sabir, M. Sadik, and L. Ladid, "Quality of Experience for Streaming Services: Measurements, Challenges and Insights," University of Luxembourg, Luxembourg, 2020.

[4] N. Banović-Ćurguz and D. Ilišević, "Mapping of QoS/QoE in 5G Networks," Mtel A.D., Banja Luka, Bosnia and Herzegovina, 2019.

[5] A. B. Poritz, "Hidden Markov Models: A Guided Tour," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 1988, pp. 7–13.

[6] F. Battisti, M. Carli, P. Paudyal, "QoS to QoE Mapping Model for Wires/Wireless Video Communication," University of Roma, Italy.

[7] E. Liotou, D. Tsolkas, and N. Passas, "A Roadmap on QoE Metrics and Models," in *Proc. 23rd Int. Conf. on Telecommunication ICT*, Thessaloniki, Greece, 2016.

[8] J. Frnda, M. Durica, M. Savrasovs, P. Fournier-Viger, J. Chun-Wei Lin, "QoS To QoE Mapping Function for IPTV Quality Assessment Based on Kohonen Map: A Pilot Study," *Transport and Telecommunication Institute*, Lomonosova, Vol. 21, no.3, 2020.

[9] W.-H. Hsu and C.-H. Lo, "QoS/QoE Mapping and Adjustment Model in the Cloud-based Multimedia Infrastructure," *IEEE Syst. J.*, 2014.

[10] P. Casas, A. D'Alconzo, F, Wamser, M. Seufert, B. Gardlo, A. Schiwind, P. Tran-Gia, R. Schatz, "Predicting QoE in Cellular Networks using Machine Learning and

in-Smartphone Measurements," Austrian Institute of Technology and University of Wurzburg, *Proc. IEEE,* 2017.

[11] A. Mekonnen, "Quality of Experience Modeling for Fixed Broadband Internet Using Machine Learning Algorithm," M.S. thesis, Dept. of Elect. and Comput. Eng., Addis Ababa Inst. Technol., Addis Ababa, Ethiopia, 2024.

[12] D. Solomon "QoE Model for Addis Ababa LTE Web Browsing Service Using Neural Network Approach," M.S. thesis, Dept. of Elect. and Comput. Eng., Addis Ababa Inst. Technol., Addis Ababa, Ethiopia, 2019.

[13] T. Abar, A.B. Letaifa, S. El Asmi, "Machine learning based QoE prediction in SDN networks," Higher School of Communication of Tunis, *IEEE*, 2017.

[14] F. C. D. Gouveia and T. Magedanz, "Quality of Service in Telecommunication Networks," vol. II. Berlin, Germany: Technical University of Berlin, Franklinstr.

[15] T. Degu, "An Assessment on the impact of virtual internet service Providers in the field of internet service delivery in Addis Ababa," St. Mary's University School of post Postgraduate Studies, May 2021.

[16] K. Mitra, C. Åhlund and A. Zaslavsky, "QoE Estimation and Prediction using Hidden Markov Models in Heterogeneous Access Network," *IEEE*, 2012.

[17] R. W. Yeung, "Historical Note on the Development of Markov Chains," Department of Mathematics, The Chinese University of Hong Kong, 2024.

[18] J. R. Norris, "Markov Chains," Cambridge, U.K.: Cambridge University Press, 1997.

[19] D. Jurafsky and J. H. Martin, "Hidden Markov Model Appendix," in *Speech and Language Processing*, 3rd ed., Draft of Jan. 12, 2025.

[20] M. Stamp, "A revealing introduction to hidden Markov models," Department of Computer Science, San Jose State University, Oct. 2018.

[21] A. Salem, "Hidden Markov Model and its Applications in Emerging Wireless Mobile Networks," M.S. thesis, Dept. of Elect. and Comput. Eng., Addis Ababa Inst. Technol., Addis Ababa, Ethiopia, Aug. 2022.

[22] S. Chandra, R. Indu, H.S Negi, N. Panwar, M. Sarda, "Hidden Markov Model - Applications, Strengths, and Weaknesses," *IEEE*, 2024.

[23] A. Diriba "Prediction of LTE Cell Degradation Using Hidden Markov Model," M.S. thesis, Dept. of Elect. and Comput. Eng., Addis Ababa Inst. Technol., Addis Ababa, Ethiopia, Aug. 2023

[24] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[25] M. Nilsson, "First order hidden markov model: Theory and implementation issues," 2005.

[26] T. M. Kodinariya and P. R. Makwana, "Review on determining number of Cluster in K-Means Clustering," *International Journal*, vol. 1, no. 6, 2013.

[27] T. Addisie, "Mobile Networks Accessibility and Retainability States Prediction Using Markov Chain," M.S. thesis, Dept. of Elect. and Comput. Eng., Addis Ababa Inst. Technol., Addis Ababa, Ethiopia, Nov. 2021.

[28] D. H. Nguyen-Le, Q. B. Tao, V.-H. Nguyen, and M. Abdel-Wahab, "A data-driven approach based on long short-term memory and hidden Markov model for crack propagation prediction," *Engineering Fracture Mechanics*, vol. 235, p. 107085, 2020.

[29] M. Chalifour, "Comparison of Sleep State Classification Performance Using Random Forests, Hidden Markov Models, and Non-homogeneous Hidden Markov Models," Department of Mathematical and Statistical Sciences, University of Alberta, 2020.

[30] A. Sharma, "Support Vector Machines (SVM) – A Complete Guide for Beginners," *Analytics Vidhya*, Oct. 25, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

[31] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, and S. Homayouni, "Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 6308–6326, Oct. 2020.

[32] B. T. Abe, O. O. Olugbara, and T. Marwala, "Experimental comparison of support vector machines with random forests for hyperspectral image land cover classification," *Journal of Earth System Science*, vol. 123, no. 4, pp. 779–790, June 2014.

[33] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[34] R. Sanyour, M. Abdullah, and S. Abdullah, "Quality of Experience in Internet of Things: A Systematic Literature Review," Information System Department, Faculty of Computing and Information Technology, King Abdul Aziz University, Jeddah, Saudi Arabia, July 2023.

[35] C. Sampaio, "Understanding SVM Hyperparameters," *Stack Abuse*, Apr. 17, 2023.

[36] J. Murel (Ph.D), "Create a confusion matrix with Python," *IBM Develope*, 2024.

[37] S. Swaminathan and B. R. Tantri, "Confusion matrix-based performance evaluation metrics," *African Journal of Biomedical Research*, vol. 27, no. 4s, pp. 4023–4031, Nov. 2024.

[38] Deepchecks Community Blog "Top Techniques for Cross-validation in Machine Learning" May 23, 2022.

[39] D. Finlay, "Transition Matrices and Markov Chains," Save My Exams, Dec. 2024. [Online]. Available: https://www.savemyexams.com/dp/maths/ib/ai/21/hl/revision-notes/statistics-and-probability/transition-matrices-and-markov-chains/transition-matrices/

[40] K. Hoffman, "Random Forest Hyperparameters Explained," *Medium*, May 23, 2020. [Online]. Available: https://ken-hoffman.medium.com/random-forest-hyperparameters-explained-8081a93ce23d

[41] A. Sharma, "Support Vector Machines (SVM) – A Complete Guide for Beginners," *Analytics Vidhya*, Oct. 25, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

[42] S. Of Zero, "Decision Trees: Bootstrap Aggregating and Bagging," *Medium*, Jun. 30, 2022. [Online]. Available: https://medium.com/@sly.of.zero/decision-trees-bootstrap-aggregating-and-bagging-8c6cf764e689