

Nicolai Hollup Brand  
Callum Gran  
Trym Hamer Gudvangen

# Predicting Underperformance In 5G Cellular Networks Using Supervised Machine Learning Based on End-User Data

Bachelor's thesis in Computer Science  
Supervisor: Olav Alseth Skundberg  
Co-supervisor: Azza Hassan Mohamed Ahmed  
May 2024



Nicolai Hollup Brand  
Callum Gran  
Trym Hamer Gudvangen

# **Predicting Underperformance In 5G Cellular Networks Using Supervised Machine Learning Based on End-User Data**

Bachelor's thesis in Computer Science  
Supervisor: Olav Alseth Skundberg  
Co-supervisor: Azza Hassan Mohamed Ahmed  
May 2024

Norwegian University of Science and Technology  
Faculty of Information Technology and Electrical Engineering  
Department of Computer Science



# Abstract

With the constant evolution of technology within cellular networks, continuous reliability poses significant problems due to a growing user-base, the complexity of technologies, and increased network density. To remedy these issues, 3GPP has proposed the Self-Organizing Network standard. State-of-the-art research demonstrates machine learning (ML) has a great potential in realizing this goal. In collaboration with SimulaMet, this thesis investigates how predictive ML models can be trained on end-user data aimed at supplementing self-healing systems.

To this end, three novel areas within predicting underperformance in 5G cellular networks were explored, using Random Forest as the baseline ML model. In this context, underperformance is defined to be abnormally high delays. Datasets were compiled using end-user data from dedicated measurement probes from eleven locations in the Oslo Metropolitan Area.

The first experiment examined changes in the performance of models for varying time granularities of active RTT measurements. It was found there is only a 7% drop off in F1-score from the best performing time granularity (one-second) to the worst, indicating efficacy in predicting 60 seconds in advance.

Further, a comparison between a centralized model and distributed models was performed. The results show the centralized model had performance improvements, outperforming ten out of the eleven distributed models with respect to F1-scores. On average, the centralized model had an average increase in F1-score of 0.1263 across all eleven nodes.

The final experiment probed the predictions from a distributed model to better understand instances of mispredictions.

This thesis concludes ML models, based on end-user data, can effectively predict underperformance up to 60-seconds in advance, giving operators a better understanding of user experience. For this purpose, a centralized model is shown to be better than distributed models. In addition, this paper has presented how tree-based models can be explained in this context.



# Sammendrag

Gitt den konstante utviklingen av teknologi innen mobile nettverk utgjør kontinuerlig pålitelighet betydelige problemer som følge av en økende brukerbase, kompleksiteten i teknologiene, og økt nettverkstetthet. For å takle disse problemene har 3GPP foreslått "Self-Organizing Network"-standarden. Nyere forskning har vist at maskinlæring (ML) har stort potensiale for å realisere dette målet. I samarbeid med SimulaMet undersøker denne oppgaven hvordan prediktive ML-modeller kan trenes på sluttbrukerdata rettet mot å supplementere "Self-Healing"-systemer.

For dette ble tre nye områder innen prediksjon av underprestasjon i 5G-nettverk utforsket, ved bruk av Random Forest som ML-modell. Her defineres underprestasjon i nettverk som unormalt høye forsinkelser i dataoverføring. Datasettet baserer seg på sluttbrukerdata fra dedikerte prober fra elleve lokasjoner i Oslo.

Det første eksperimentet undersøkte endringer i modellenes ytelse for varierende tidsgranulariteter av aktive RTT-målinger. Det ble funnet at det kun er en minskning på 7% i F1-score fra den beste tidsgranulariteten (ett sekund) til den verste (60 sekunder), som indikerer effektivitet i å predikere underprestasjon 60 sekunder i forveien.

Videre ble det sammenlignet ytelse mellom en sentralisert modell og distribuerte modeller. Resultatene viser at den sentraliserte modellen hadde bedre ytelse og presterte bedre enn ti av de elleve distribuerte modellene med hensyn til F1-score. I gjennomsnitt hadde den sentraliserte modellen en økning i F1-score på 0.1263 i forhold til de distribuerte modellene.

Det siste eksperimentet undersøkte prediksjonene fra en distribuert modell for å bedre forstå hvor og hvorfor den gjetter feil.

Denne oppgaven viser at ML-modeller, basert på sluttbrukerdata, kan effektivt forutsi underprestasjon opptil 60 sekunder i forveien, noe som gir operatører en bedre forståelse av brukeropplevelse i nettverk. Til dette formålet vises det at en sentralisert modell er bedre enn distribuerte modeller. I tillegg har dette arbeidet presentert hvordan tre-baserte modeller kan forklares.

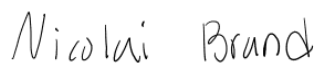




# Preface

This research project is the concluding thesis for the bachelor program *Computer Science* at the Norwegian University of Science and Technology (NTNU) and is written in collaboration with the research institution Simula Metropolitan Center for Digital Engineering AS (SimulaMet). The motivation for this project is based on the ever increasing complexity of cellular networks, and the subsequent cost of maintenance and infrastructure. In this, thesis we will explore how supervised machine learning models can be employed to predict the occurrence of underperformance in 5G cellular networks.

We want to thank our supervisor Olav Alseth Skundberg for offering important insights and great interest in our thesis. An additional thanks goes to NTNU's High Performance Computing Group for allowing us to use their hardware, which has been crucial when performing research. Finally, we would like to show our sincerest gratitude to our co-supervisor at SimulaMet, Azza Hassan Mohamed Ahmed, for making this project possible and providing invaluable guidance throughout our research.



---

*Nicolai Hollup Brand*



---

*Callum Gran*



---

*Trym Hamer Gudvangen*

Trondheim, Tuesday 21<sup>st</sup> May, 2024



# Assignment Details

The following is the original assignment statement received from SimulaMet before work on the project commenced:

*"Build a machine learning based diagnosis system as a part of the self-healing functionality in mobile networks. The proposed system uses network condition indicators such as key performance indicators and performance management counters for network condition diagnosis."*

For reasons outlined later in the thesis, the task pivoted. Originally a diagnosis system was the main focus, but as circumstances changed, a predictive system was researched instead. From this, several interesting questions emerged and were explored, even though they fall somewhat outside the scope of the original assignment statement.



# Contents

<b>Abstract</b>	<b>i</b>
<b>Sammendrag</b>	<b>iii</b>
<b>Preface</b>	<b>v</b>
<b>Assignment Details</b>	<b>vii</b>
<b>Contents</b>	<b>ix</b>
<b>Figures</b>	<b>xiii</b>
<b>Tables</b>	<b>xvii</b>
<b>Acronyms</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 The Research Questions . . . . .	3
1.3 Structure . . . . .	5
<b>2 Theory</b>	<b>7</b>
2.1 Cellular Communication . . . . .	7
2.1.1 Cellular Networks . . . . .	7
2.1.2 Service-Level Agreement . . . . .	8
2.1.3 Network Performance . . . . .	8
2.1.4 Perceived Network Performance . . . . .	9
2.1.5 Network Faults . . . . .	10
2.1.6 Self-Organizing Networks . . . . .	10
2.2 Statistical Methods . . . . .	12
2.2.1 Classification . . . . .	12
2.2.2 Outlier Detection . . . . .	12
2.2.3 Kernel Density Estimation . . . . .	13
2.3 Machine Learning Preliminaries . . . . .	13
2.3.1 Dataset . . . . .	14
2.3.2 Validation . . . . .	19
2.3.3 ML Model Performance . . . . .	23
2.3.4 Model Optimization . . . . .	25
2.3.5 Explainability and Interpretability . . . . .	26

2.3.6 Supervised Machine Learning Algorithms . . . . .	28
2.3.7 Deep Learning . . . . .	32
<b>3 Method</b>	<b>35</b>
3.1 Planning and Deciding the Research . . . . .	35
3.2 Research Approach . . . . .	37
3.2.1 Scientific Method . . . . .	37
3.2.2 Research Design . . . . .	37
3.3 Development Approach . . . . .	40
3.3.1 Choice of Technologies . . . . .	40
3.3.2 Ensuring Code Quality . . . . .	42
3.3.3 Documentation . . . . .	43
3.3.4 Progress Reports . . . . .	43
3.3.5 Standardized Communication . . . . .	44
3.4 Preliminary Setup . . . . .	45
3.4.1 Data Collection . . . . .	46
3.4.2 Initial Exploratory Analysis . . . . .	47
3.4.3 Initial Dataset Selection . . . . .	48
3.4.4 Merging the Dataset . . . . .	48
3.4.5 Data Wrangling . . . . .	49
3.4.6 Feature Engineering . . . . .	50
3.4.7 Exploratory Analysis . . . . .	53
3.4.8 Defining Underperformance in the Network . . . . .	57
3.4.9 Addressing Data Imbalance . . . . .	58
3.4.10 Model Choice . . . . .	60
3.4.11 Validation Metrics . . . . .	61
3.5 Conducting Research Experiments . . . . .	61
3.5.1 Data Granularity Experiment . . . . .	62
3.5.2 Centralized vs. Distributed Learning Experiment . . . . .	62
3.5.3 Understanding Model Output . . . . .	64
3.5.4 Tackling explainability of the model . . . . .	64
<b>4 Results</b>	<b>67</b>
4.1 Model Comparison . . . . .	67
4.1.1 Grid-Search . . . . .	67
4.1.2 Comparison Using Best Estimator . . . . .	68
4.2 Time Granular Measurements . . . . .	72
4.3 Centralized vs. Distributed Models . . . . .	76
4.3.1 General Performance . . . . .	76
4.3.2 Node-specific Performance . . . . .	78
4.4 Understanding Model Predictions . . . . .	80
4.4.1 General Findings . . . . .	81

4.4.2 Investigating Mispredictions . . . . .	84
4.4.3 Effect of Features . . . . .	85
<b>5 Discussion</b>	<b>91</b>
5.1 Comparing the ML Model Types . . . . .	91
5.1.1 Choice of Metrics . . . . .	91
5.1.2 On Grid-Search . . . . .	91
5.1.3 Analyzing Model Performance . . . . .	92
5.1.4 Chosen Model . . . . .	93
5.1.5 Evaluating the Sampling Methods . . . . .	94
5.2 Time Granular Measurements . . . . .	95
5.2.1 Interpreting Results for Varying Granularities . . . . .	95
5.3 Centralized vs. Distributed Models . . . . .	97
5.3.1 Interpreting the Results . . . . .	97
5.3.2 Benefits of a Centralized Model . . . . .	98
5.3.3 Benefits of Distributed Models . . . . .	98
5.4 Understanding Model Predictions . . . . .	99
5.4.1 Trends in Underperformance . . . . .	99
5.4.2 Causes of Mispredictions . . . . .	99
5.4.3 Explainability of Model Predictions . . . . .	100
5.5 Supplementing Self-Healing Systems with End-User Data . . .	102
5.6 Reflection on Methodology . . . . .	103
5.6.1 Reproducibility of the Research . . . . .	103
5.6.2 Validity of Research . . . . .	103
5.6.3 Setbacks . . . . .	104
<b>6 Conclusion</b>	<b>107</b>
<b>7 Societal Impact</b>	<b>109</b>
<b>8 Further Work</b>	<b>111</b>
<b>Bibliography</b>	<b>113</b>
<b>Appendix A Interview with Experts</b>	<b>i</b>





# Figures

1.1 Overview of Research Area. . . . .	3
2.1 Image of a 5G Cell Tower. (Photo by Authors). . . . .	8
2.2 Overview of Cell Degradation Management Within the Self-Healing Module. The "Solution Deployment" Block Encapsulates the Healing Process Post Root Cause Analysis. Diagram Inspired by Hämäläinen et al., 2011. . . . .	11
2.3 Quadratic Model Fitted on Noisy Quadratic Generated Dataset. . . . .	14
2.4 General Method For Performing Data Wrangling. . . . .	15
2.5 Example of One-Hot Encoding of Day-of-the-Week. . . . .	16
2.6 The Confusion Matrix. . . . .	20
2.7 Receiver Operating Characteristic Graph. The Red Dot in The Figure Represents the Model in Figure 2.6. . . . .	23
2.8 Flowchart Describing The Grid-Search Process. Inspired by Uddin et al., 2021. . . . .	26
2.9 Diagram Depicting Various Machine Learning Categories Taken from Sarker, 2021. . . . .	29
2.10 Logistic Regression Applied to an Example Dataset Using an Arbitrary Threshold Value. . . . .	30
3.1 Research Method Model. This Model is Inspired by Hevner (2007, Fig. 1, p. 3) . . . . .	38
3.2 Slurm Script Used For Running Batch Jobs on IDUN. . . . .	42
3.3 Structure of Google Drive. . . . .	43
3.4 The Team's Kanban Board on Jira . . . . .	44
3.5 The Preliminary Development for The ML Pipeline. . . . .	45
3.6 Map Visualization of NorNet from RobusteNett Website. . . . .	46
3.7 Plots of Constant Values Found Initial Exploratory Analysis. . . . .	47
3.8 How Passive Metadata Events Were Merged into the One-Second Measurements to Form a Unified Dataset. . . . .	49
3.9 Data Engineering Population to Node. . . . .	51
3.11 Box Plot overlaid with KDE for Population based on Node. . . . .	51
3.10 KDE for Population based on Node with Minima. . . . .	52
3.12 Boxplot for RTT of Five Nodes over One Week. . . . .	54
3.13 Boxplot of Individual Node RTT with Cut-Off for Underperformance. . . . .	55

3.14Cumulative Distribution Function for RTT of Five Nodes over One Week. . . . .	56
3.15Pearson Correlation Matrix for Features. . . . .	56
3.16Class Distribution Before Sampling. . . . .	58
3.17Class Distribution After Undersampling. . . . .	59
3.18Class Distribution After Oversampling. . . . .	59
3.19Flow Chart of How the Experiments Were Carried Out. . . . .	61
3.20Diagram Representing The Network Landscape with Differing Models. . . . .	63
3.21Explainability Categorization of Chosen Models. . . . .	64
4.1 Scatter Plots Displaying Precision vs Recall for Each Model-Compare Measurement. Top Right is Better. . . . .	70
4.2 Scatter Plots displaying Precision vs Recall for each Granularity Measurement. Top Right is Better. . . . .	73
4.3 Scatter Plots Displaying TP-Rate vs FN-Rate for each Granularity Measurement. Top Left is Better. . . . .	74
4.4 Bar Chart Comparing F1 Score for Each Granularity Measurement. Higher is Better. . . . .	75
4.5 Bar Chart Comparing AUPRC Score for each Granularity Measurement. Higher is Better. . . . .	75
4.6 Scatterplot Showing Disk Space Against Compute Time per Analysis for the Centralized and Distributed Models. . . . .	77
4.7 Scatterplot Showing Recall vs. Precision for the Centralized and Distributed Models. . . . .	77
4.8 Scatterplot Showing Precision Against Recall for the Centralized Model, Validated against Nodes Individually. . . . .	78
4.9 Scatterplot Showing F1-score per Node with Centralized and Distributed Model. . . . .	79
4.10Scatter Plots Displaying TP-Rate and FP-Rate. . . . .	80
4.11Number of Underperformance Cases per Minute for the Entire Three Month Period. You may need to zoom in to see the figure text. . . . .	81
4.12Number of Underperformance Cases per Minute for the First Day. . . . .	81
4.13Fraction of Underperformance per Hour of the Day. . . . .	83
4.14Fraction of Underperformance per Day of the Week for the Entire Centralized Dataset. . . . .	84
4.15Mean RTT Value for all RTT Features Categorized into Various Classes. . . . .	85
4.16Feature Importance of the Model for Node 4144 using MDI. . . . .	86
4.17Feature Importance of the Model for Node 4144 using PaP. . . . .	87

4.18SHAP Global Feature Importance of the Model for Node 4144.	87
4.19SHAP Local of True Positive Outputs for the Distributed Model.	88
4.20SHAP Local of False Positive Outputs for the Distributed Model.	89
4.21SHAP Local of True Negative Outputs for the Distributed Model.	90
4.22SHAP Local of False Negative Outputs for the Distributed Model.	90



# Tables

2.1 Major Methods of Oversampling. . . . .	18
2.2 Major Methods of Undersampling . . . . .	18
3.1 The Criteria Employed When Deciding What Direction to Take the Thesis. . . . .	36
3.2 Strengths and Weaknesses of Chosen Research Design. . . . .	40
3.3 Collection of Most Valuable Third-Party Python Libraries. . . . .	41
3.4 Hardware Specification Used to Train and Test Models. . . . .	41
3.5 Metrics Chosen After Initial Dataset Selection. . . . .	48
3.6 Valid Feature Intervals. . . . .	50
3.7 List of Features in Dataset. . . . .	53
3.8 Hyperparameter Space for Each Model. . . . .	60
4.1 The Best Values for Each Hyperparameter After Grid-Search. . . . .	68
4.2 Performance Metrics for the Best Estimator for Each Model. . . . .	68
4.3 Performance Metrics for the Best Estimator for Each Model. . . . .	71
4.4 Performance Metrics of RF with Various Resampling Methods. . . . .	71
4.5 Performance Metrics for the Random Forest Model Across Dif- ferent Time Granularities. . . . .	72
4.6 Centralized vs. Distributed Model Results. . . . .	76
4.7 Centralized Model Performance per Node. . . . .	78
4.8 Passive Measurement Features Categorized into TP and FN Classes. . . . .	84



# Acronyms

<b>3GPP</b>	3rd Generation Partnership Project.
<b>AdaBoost</b>	Adaptive Boosting.
<b>AUC</b>	Area Under Curve.
<b>AUPRC</b>	Area Under Precision-Recall Curve.
<b>CatBoost</b>	Categorical Boosting.
<b>dBm</b>	decibel-milliwatts.
<b>DSR</b>	Design Science Research.
<b>FN</b>	False Negative.
<b>FP</b>	False Positive.
<b>HPC</b>	High Performance Computing.
<b>IQR</b>	Interquartile Range.
<b>KDE</b>	Kernel Density Estimation.
<b>KPI</b>	Key Performance Indicator.
<b>LightGBM</b>	Light Gradient Boosting Machine.
<b>LTE</b>	Long Term Evolution.
<b>MDI</b>	Mean Decrease in Impurity.
<b>ML</b>	Machine Learning.
<b>NNE</b>	Nornet Edge.
<b>PaP</b>	Permute-and-Predict.
<b>RF</b>	Random Forest.
<b>ROC</b>	Receiver Operating Characteristic.
<b>RSRP</b>	Reference Signal Received Power.
<b>RSRQ</b>	Reference Signal Received Quality.
<b>RSSI</b>	Received Signal Strength Indicator.
<b>RTT</b>	Round Trip Time.
<b>SHAP</b>	SHapley Additive exPlanations.
<b>SimulaMet</b>	Simula Metropolitan Center for Digital Engineering AS.
<b>SLA</b>	Service-Level Agreement.
<b>SON</b>	Self-Organizing Network.
<b>SVC</b>	Support Vector Classification.
<b>TN</b>	True Negative.
<b>TP</b>	True Positive.
<b>XAI</b>	Explainable AI.
<b>XGBoost</b>	Extreme Gradient Boosting.





# 1. Introduction

The self-healing component of the Self-Organizing Network (SON) specification works to automatically identify, diagnose, and resolve faults in cellular networks. Due to the technical and financial complications from new technologies (T. Chen et al., 2014), self-healing systems are viewed as essential to alleviate the workload for expert operators (Asghar et al., 2018). However, while most self-healing systems utilize intra-network data in isolation, the paper Turner et al., 2012 argues these systems may be unable to representatively capture every facet of the user's experience. According to Asghar et al., 2018, a possible next step in self-healing is a proactive system to predict user load in the network.

This research aims to investigate how end-user data can be used to supplement self-healing systems through predicting the occurrence of network underperformance. The following chapter will therefore first go over the foundational context for this thesis. From there, the research questions are formulated and motivated. Finally, the structure of the thesis will be presented.

## 1.1 Background

While Long Term Evolution (LTE)-networks promise high efficiency and low latency (European Telecommunications Standards Institute, 2019), continuous reliability poses a significant problem (Siddiqi et al., 2019). The main challenges stem from a growing user-base (K.-M. Chen et al., 2019), emerging complexity due to integration of new technology, and increased network density (Asghar et al., 2018). To combat these issues, 3rd Generation Partnership Project (3GPP) has proposed the self-healing standard under the SON specification. The self-healing standard aims at the automatic detection, diagnosis, and resolution of performance issues of cellular networks (Hämäläinen et al., 2011).

One meta-study on self-healing found between 23-26% of the total revenue of telecommunication operators goes to operational expenditures and concludes self-healing systems are "no longer a luxury, but rather a necessity" (Asghar et al., 2018). To alleviate the workload and expenses surrounding network experts, research on automatic fault detection within the self-healing module has become popular (Szilagyi & Novaczki, 2012) (Mdini et al., 2020) (K.-M. Chen et al., 2019).

Telecommunication operators have as a goal to deliver high-quality service to the end-user. Therefore, to improve the end-user's experience, network operators constantly monitor the network to quickly identify and resolve any occurring issues (Hämäläinen et al., 2011). However, understanding and monitoring what the end-user actually experiences poses an issue for the telecommunication operators.

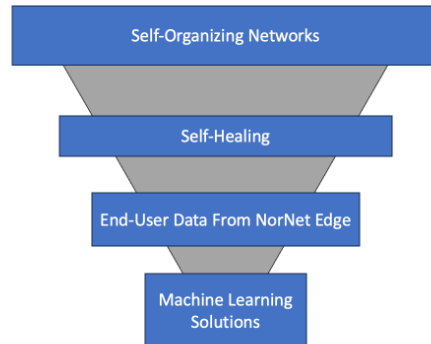
In an attempt to better understand the user-side of cellular networks, the Nornet Edge (NNE) project was established. The NNE platform measures end-user data from cellular networks all over Norway (Ahmed et al., 2021). These measurements are done by continually probing the network with over 400 fixed and mobile devices that connect to the cellular network (Kvalbein et al., 2014).

Many papers on self-healing, such as K.-M. Chen et al., 2019 and Szilagyi and Novaczki, 2012, utilize simulated intra-network data. However, simulations are intrinsically simplified models of reality and therefore cannot capture all facets of a real network's state. Instead, other papers use closed-source datasets from operators, yielding more realistic results while limiting the reproducibility and further advancement of the research. Moreover, there were only a few papers investigating how real end-user data fits into the larger picture. One paper suggests purely intra-network-based systems may be unable to representatively capture the user's experience (Turner et al., 2012). It is important to fill this gap in research because it has the potential to enable a more holistic understanding of end-user experience in the network, which is a prerequisite for quicker and better self-healing solutions.

According to Asghar et al., 2018, a possible next step in self-healing is a proactive system to predict user behavior in the network. The dataset gathered through NNE lends itself nicely to investigating such a predictive system using end-user data.

Heuristic and simple statistical approaches have been utilized in previous attempts to automate self-healing (Szilagyi & Novaczki, 2012). However, due to the volume and complex relationships of data, Machine Learning (ML) based systems have been proposed for self-healing applications, experiencing varying degrees of success (Asghar et al., 2018).

The datasets compiled through NNE may pose similar problems with complexity. Prior research highlights the dependency of statistical solutions on domain expertise and manual updating of parameters (K.-M. Chen et al., 2019). Due to the lack of such competency, this thesis decides to not pursue complex statistical modelling as a solution for this problem. Instead, the research will be conducted surrounding the use of ML systems.



**Figure 1.1:** Overview of Research Area.

Network providers are naturally hesitant with fully replacing operators with automatic ML systems. Trust must therefore be built between the SON implementation and network experts. To successfully coordinate this interaction, ML models must provide a certain degree of explainability before integration becomes viable. It must, in other words, be possible to reason about the output of a model, given its architecture, state, and input.

## 1.2 The Research Questions

With a basis in the identified gaps in the literature, current solutions for improving user-experience, and the access to end-user data, the main research question for this thesis is:

**“How can a predictive supervised machine learning model handle end-user data to supplement self-healing systems based on 3GPP’s specifications?”**

Intra-network probe measurements may contain insightful information but is restricted to logging symptoms of network failure and unable to directly assess the impact on the user (Turner et al., 2012). The goal of this research is therefore to complement self-healing systems with data reflecting how the user perceives the performance of the cellular network. The major challenges with end-user measurements is the need for substantial disk storage space and increased network load due to continuously pinging to obtain active measurements.

To tackle these challenges, this research paper will start off by investigating how ML systems can predict network underperformance ahead of time based on the NNE end-user data in the hopes of reducing the frequency of active measurements. An interesting sub-question from the perspective of both NNE, who collects the data, and the telecommunication operators becomes:

**1. "How does modulating the granularity of active end-user measurements affect model efficacy when predicting cellular network underperformance?"**

When predicting network underperformance across each NNE probe, it is unclear whether a separate ML model trained for each distinct device is necessary or if a centralized model would be more effective. Furthermore, this architectural choice would not only affect performance but also practical details surrounding implementation and deployment. The next sub-question to explore is therefore:

**2. "What are the trade-offs in deploying distributed machine learning models for each NorNet Edge node versus a centralized model for predicting cellular network underperformance?"**

Finally, supplements to self-healing solutions are only useful when integrated into a real-world scenario. A significant aspect of the solution is therefore assessing the ease of integration, usability, and compatibility with the needs of telecommunication operators. An interview was held with the Norwegian telecommunication operator Telenor to establish a broader understanding of the current problems and solutions. In the context of ML systems, a hard-requirement for integration is an acceptable degree of explainability. Telecommunication operators are reluctant to incorporate solutions that function as "black boxes" into their networks. Thus, the clarity and transparency of an ML solution must be taken into consideration:

**3. "What considerations need to be taken regarding explainability and interpretability before integration of an automatic network solution becomes viable?"**

This thesis is written in collaboration with Simula Metropolitan Center for Digital Engineering AS (SimulaMet). SimulaMet was established in 2018 as a non-profit research organization between the Simula Research Laboratory and Oslo Metropolitan University. The organization conducts research

within communication systems, ML and data science, and information technology management.

## 1.3 Structure

The thesis is structured into the following chapters to provide a comprehensive overview of the subject matter:

- **Chapter 1: Introduction** - This chapter presents the context and background of the thesis, explaining the reasoning and importance of this research. The introduction also outlines the general structure of the thesis and describes the research questions.
- **Chapter 2: Theory** - This chapter describes the theoretical framework necessary to understand the thesis.
- **Chapter 3: Method** - The method chapter details the research and development methodology conducted during the project.
- **Chapter 4: Results** - In the results chapter, the findings of the research will be presented.
- **Chapter 5: Discussion** - This chapter will analyze and interpret the results from the preceding chapter. In addition, the significance of the results will be explored in relation to the research questions from the introduction.
- **Chapter 6: Conclusion** - This chapter provides conclusions for the work based on the method, results, and discussion, answering the research questions stated in the introduction.
- **Chapter 7: Societal Implications** - This chapter briefly reflects upon the broader societal impact of the research and potential ramifications of the results on a societal scale.
- **Chapter 8: Further Work** - This chapter explores future possibilities and effects of the research conducted in this thesis.
- **Bibliography** - The bibliography will outline all sources cited in the thesis.
- **Appendix** - The appendix will include additional attachments referred to in the thesis.



## 2. Theory

This chapter aims to give the reader the necessary theoretical background to understand the methods used further in this thesis. To ensure this, the chapter is split into three separate parts. First, the reader will learn essential information about mobile wireless networks and their standards, with a specific focus on cellular technologies. Further, the chapter will introduce some statistical methods, continued thereon by ML theory.

### 2.1 Cellular Communication

The following section will cover essential network concepts, terminology, and standards that are vital to understanding the problem addressed in this thesis. This knowledge will lay the foundation for understanding why the problem is relevant and what factors need to be taken into account when creating an effective solution.

#### 2.1.1 Cellular Networks

Wireless networks are computer networks that transmit data wirelessly through the air using radio frequencies. Mobile wireless networks are a subset of wireless networks that offer connectivity to mobile devices. Such networks allow for access to the network even when the device is moving between different geographic locations (Kurose & Ross, 2013). The most common type of mobile wireless networks are cellular networks such as 3G, 4G and now the emergence of 5G and later 6G technologies. Cellular networks work by splitting a geographical area into smaller areas, known as cells. These networks support multiple services such as internet and voice calls (Kurose & Ross, 2013).

Before going deeper into the topic of cellular networks, it is necessary to understand some of the key components that constitute them.

#### **Transceiver**

A transceiver is a device that can both transmit and receive data. For wireless networks, transceivers send and receive radio waves using an antenna (Luzzatto & Haridim, 2016).

### Base Station

The base station is a critical component of cellular networks (Kurose & Ross, 2013). In essence, the base station is a fixed transceiver that facilitates wireless communication between connected devices through a wireless host. A cell tower is an example of a base station.



**Figure 2.1:** Image of a 5G Cell Tower. (Photo by Authors).

#### 2.1.2 Service-Level Agreement

A Service-Level Agreement (SLA) is a contract between a network operator and an end-user. This agreement describes level of service (performance) expected by the operator within quality and reliability. Levels of metrics such as bandwidth, latency, packet loss, uptime, and jitter could be defined in such an agreement (Akbari-Moghanjoughi et al., 2019).

#### 2.1.3 Network Performance

Due to the dynamic nature of cellular networks, monitoring and maintaining network health is necessary for ensuring high quality user experience (Hämäläinen et al., 2011). The network state being *healthy* in this context is defined as the network performing at its expected performance or better. *Network underperformance*, on the other hand, is defined as the contrary to a healthy network, meaning when the performance is significantly worse than the expected performance. Hämäläinen et al., 2011 calls this



*network degradation* while other sources refer to it as the network being in a *faulty* state. This thesis will use the term network underperformance.

Network performance is usually audited using intra-network data, but it is possible to utilize end-user data, or a combination of both (Kvalbein et al., 2014). This data is used to calculate metrics that provide insight into various aspect of network quality, health and operation. These metrics serve as important Key Performance Indicators (KPIs), and the process of evaluating network health often involves monitoring deteriorations in KPIs.

It is important to note that a deterioration in a KPI might not necessarily stem from the occurrence of a fault in a network. For instance, deteriorations in a KPI can be caused by unusual, but not faulty, user behavior. Therefore, it is vital to clearly distinguish between network underperformance and the actual occurrence of a network fault.

Below is a collection of KPIs directly attainable or otherwise calculated from end-user data.

### **Collection of Important end-user KPIs**

*Round Trip Time (RTT)* measures the time it takes for a network packet to travel from source to destination, and back. RTT can be interpreted as a measurement of responsiveness. A lower RTT value generally implies better responsiveness.

*Received Signal Strength Indicator (RSSI)*, *Reference Signal Received Quality (RSRQ)* and *Reference Signal Received Power (RSRP)* are important end-user KPIs specifically for the LTE standard (Afroz et al., 2015). As the name suggests, RSSI measures cellular signal strength. RSSI values are expressed in decibel-milliwatts (dBm) ranging from 0, perfect signal, to no signal which is at -120 dBm (Lazar et al., 2023). RSRP is a more refined signal strength indicator as it specifically measures the power of LTE signals over the full bandwidth. RSRQ provides further information about the quality of the received signal.

#### **2.1.4 Perceived Network Performance**

While the KPIs in 2.1.3 provide a quantifiable measure of network performance, the perceived performance by the end-user may be different. In discussions about perceived network quality, terms like Quality of Service and end-user experience are commonly used. However, as brought up in Kvalbein et al., 2014, these terms are subjective in nature and problematic to quantify.

For perception of network performance, context becomes a significant factor. For instance, the end user's individual expectations of quality, along with data intensity of the application they are using, can result in multiple users having considerably different experiences on the same network.

### **2.1.5 Network Faults**

#### **Sources of Network Faults**

Faults in cellular networks occur due to their exceptional size and complicated internals (Hämäläinen et al., 2011). These issues will only become more frequent as cellular networks continue to grow in size, density and complexity (Asghar et al., 2018). Some examples of common network faults is antenna tilt, configuration errors, network interference, and more.

#### **Identifying and Correcting Network Faults**

Today, identifying network faults are mostly performed manually. Certain tasks are automated, but troubleshooting engineers are permanently analysing data from the network attempting to identify faults (Hämäläinen et al., 2011). The manual nature of this process can result in significant delays before a fault can be rectified (Hämäläinen et al., 2011). To enhance the process of detecting and diagnosing faults, the notion of SON has been introduced (European Telecommunications Standards Institute, 2019).

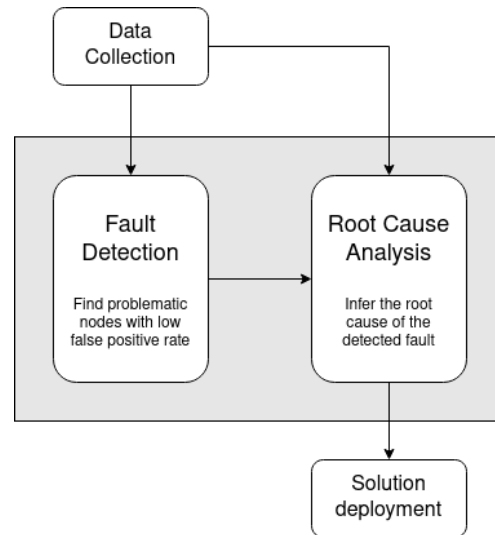
### **2.1.6 Self-Organizing Networks**

A SON is defined by three properties: self-configuration, self-optimization, and self-healing (European Telecommunications Standards Institute, 2019). The purpose of a SON is to reduce human interaction and intervention; in other words, the network should be partly autonomous. The three properties mentioned above are essential for scaling down operational dependence. For this thesis, self-healing will be the main focus, and will therefore be described more in depth.

#### **Self-Healing**

Self-healing is the property of a network to detect, diagnose, and resolve faults within its system automatically. As outlined in 2.1.5, faults are not uncommon and can arise from a myriad of conditions; the severity can also vary significantly. A key idea behind self-healing is to "mitigate faults which could be solved automatically by triggering appropriate recovery actions" (3GPP, 2022). To achieve this, the self-healing module requires

components for detecting the occurrence of a fault, diagnosing the fault, and applying corrective measures.



**Figure 2.2:** Overview of Cell Degradation Management Within the Self-Healing Module. The “Solution Deployment” Block Encapsulates the Healing Process Post Root Cause Analysis. Diagram Inspired by Hämäläinen et al., 2011.

The main focus of self-healing research has been in “cell degradation management” (Van den Berg et al., 2008). The term “cell degradation” broadly encapsulates all categories of faults that can occur in a cellular network cell. Cell degradation management can broadly be split up into two modules: the cell degradation detection module (also known as the fault detection module), and the root cause analysis module.

The responsibility of the cell degradation module is to constantly monitor the network looking to identify underperforming cells. If a cell is deemed underperforming, it is forwarded through to the root cause analysis module whose task is to identify the category of fault, if one exists, so corrective measures can be applied (Hämäläinen et al., 2011).

## SON Implementations

A key aspect of SON is its autonomous nature. Contemporary SON implementations, especially the self-healing module, utilize expert, statistical and ML systems (Asghar et al., 2018). The next two sections will cover the statistical and ML methods and techniques required to develop prediction systems that can be incorporated into larger SON systems.

## 2.2 Statistical Methods

Statistics is at the heart of any prediction system as it provides the tools and techniques necessary to analyse and understand data. This provides a framework to uncover relationships and dependencies in the data which can be used to make inferences and predictions.

### 2.2.1 Classification

Classification is a statistical concept used to categorize data and is used for making predictions about the categorical outcomes of new data based on old data. This is done by creating boundaries that form decision areas, which in turn decide what and how a data point is categorized (Marques de Sá, 2003). A trivial example of classification is binary classification with a single condition on one variable. If we have the decision boundary:

$$\{x : x^T \hat{\beta} = 0.5\}$$

where  $x^T \hat{\beta}$  is the boundary, then elements that have a value  $x > 0.5$  will be classified as category *A* and values  $x \leq 0.5$  as category *B* (Hastie et al., 2009).

### 2.2.2 Outlier Detection

Outlier detection is the identification of data points that differ significantly from the majority of data. Identification of these points is of interest to remove any data that does not follow general trends of the data, although these data points could also be looked at on their own to give insights or discoveries about the data (Seo, 2006).

#### Interquartile Range

Tukey's method, better known as Interquartile Range (IQR), is a measure that quantifies the spread around the middle 50% of a dataset. IQR is not sensitive to extreme outliers making it useful for identifying outliers in skewed dataset or when the data is not normally distributed. The method itself is based on the difference between the first and third quartile of a dataset, where the first quartile is the 25th percentile and the third quartile is the 75th percentile. The formula for IQR is therefore the following:

$$IQR = Q3 - Q1$$

When used for outlier detection, the  $1.5 \times IQR$  rule is usually followed. The rule is based on symmetric distribution and non-outlier data points  $x$  are

found by:

$$Q_1 - 1.5 \times IQR \leq x \leq Q_3 + 1.5 \times IQR$$

(Seo, 2006).

### 2.2.3 Kernel Density Estimation

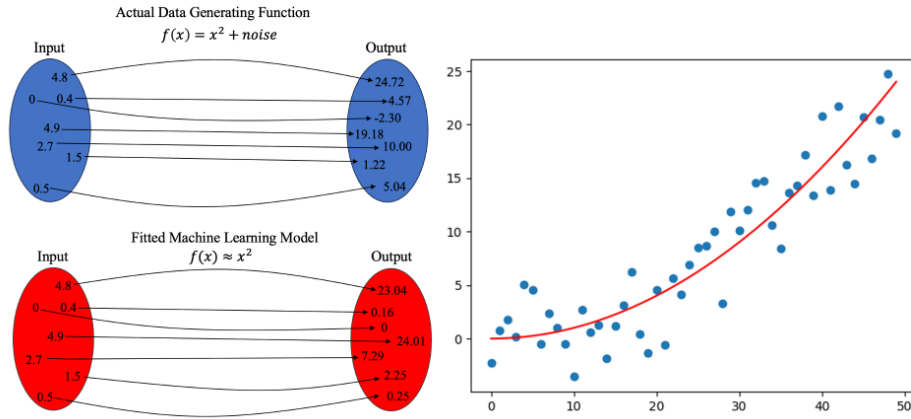
Kernel Density Estimation (KDE) is a statistical analysis method used in visualization to estimate density function of a variable based on data points. It uses kernel smoothing techniques to create a distribution graph. The degree of smoothing can be altered based on the choice of the kernel and the bandwidth parameter  $h$ . A Gaussian kernel is commonly used, but there are others, each having their own characteristics influencing the density estimate. The bandwidth  $h$  controls the degree of smoothing. When  $h$  is larger, the results are smoother, and vice versa for a smaller  $h$ , but smaller  $h$  values can introduce noise. Mathematically a KDE can be represented as:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

where  $\hat{f}(x)$  is the estimated density function,  $n$  is the number of data points,  $x_i$  are the data points,  $K$  is the kernel function, and  $h$  is the bandwidth (Węglarczyk, Stanisław, 2018).

## 2.3 Machine Learning Preliminaries

The goal of ML is to approximate the underlying function that maps inputs to outputs within a dataset. Given such a function, ML models can approximate the correct output given new inputs not encountered previously. Ultimately, all ML problems boil down to fitting models as accurately as possible to the original data distribution for a given problem.



**Figure 2.3:** Quadratic Model Fitted on Noisy Quadratic Generated Dataset.

### 2.3.1 Dataset

The dataset is the backbone of any ML system (Maharana et al., 2022). It is therefore paramount to not only collect enough information but to ensure high quality data. What constitutes as high quality is dependent on the problem, but a few shared requirements according to Fürber, 2016 are:

1. availability
2. correctness
3. clarity (i.e. minimal noise)
4. uniqueness

After collecting data from relevant sources, it becomes important to consolidate the data into one dataset. The completed dataset can then be used to train and test the ML model. The whole process from raw data to a complete dataset can vary greatly; however, all ML pipelines perform some form of data pre-processing. The two major steps while preparing data is data wrangling and feature engineering (Maharana et al., 2022).

#### Data wrangling

Data wrangling, or data munging, encompasses several crucial steps to transform raw data into a format that is easier to analyze and visualize. Qlik, 2024 defines data wrangling as a six step process, although based on other sources like Bogatu et al., 2018, the following five will be defined

for this thesis:

**Data Discovery:** Understanding the available data, identifying data quality issues, and determining the data's initial state and how to handle it.

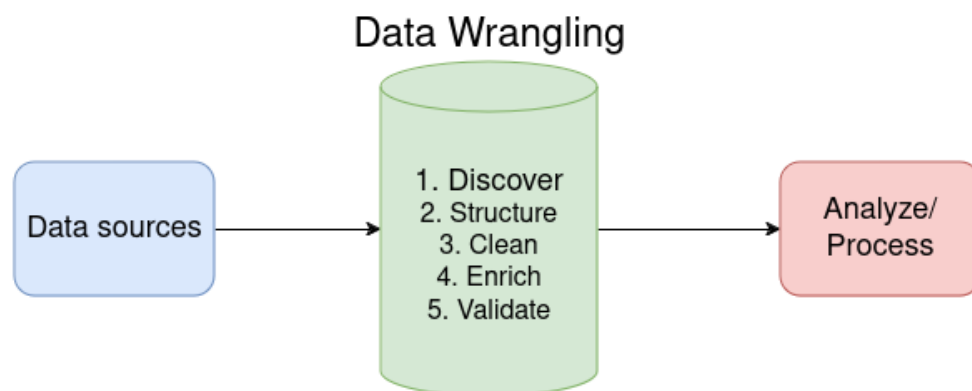
**Data Structuring:** Converting unstructured data into a structured format, which makes it easier to manipulate and analyze. This involves organizing the data into comprehensible formats such as CSV.

**Data Cleaning:** This step involves removing or correcting anomalies, filling missing values, and addressing outliers in the data to improve its quality and accuracy.

**Data Enriching:** Augmenting the existing data with additional sources can provide new insights or enhance the data's detail, which is especially useful in analysis.

**Data Validating:** Applying tests to ensure the data's accuracy and consistency according to the predefined standards before it's used for further analysis.

These steps together play an important role in preparing data for analysis and processing, which will ensure that ML models have less noise and perform better (Bogatu et al., 2018). Figure 2.4 abstractly details this process.



**Figure 2.4:** General Method For Performing Data Wrangling.

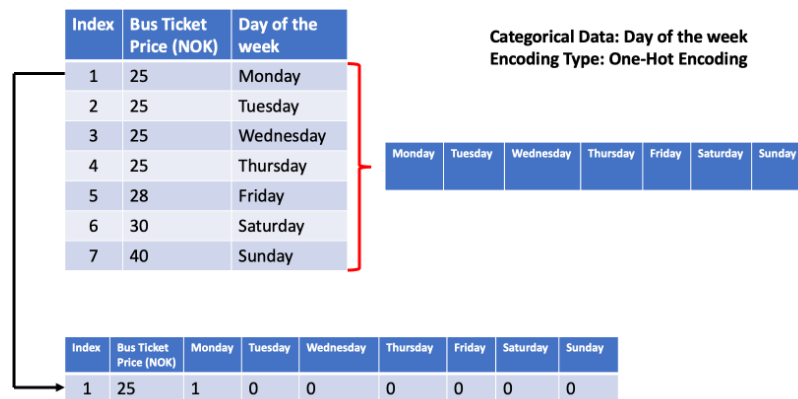
### Feature Engineering

Feature engineering is the act of generating desired features from other features. This process may, in turn, remove correlational features, add completely new information to the dataset, and simplify the training process for the model (Whitfield, 2023).

**Categorical data handling** Categorical data is data whose values contain labels rather than numerical values. A feature representing the day of the week (Monday, Tuesday, etc.) is a good example of this. While this makes sense to humans, computing devices do not have direct semantic understanding, meaning the context behind the labels are lost. Most ML algorithms therefore need the input data to come in the form of numerical data.

To encode the day of the week into numerical data, a straightforward solution would be to assign sequential unique numbers to each day starting at 1. This is called integer-encoding (Potdar et al., 2017). However, this approach has one major flaw – it implies an inherent order between the categories. For example, an ML model may be misled to believe Wednesday is somehow “greater” than Monday, or that Tuesday is the average of Monday and Wednesday.

Instead, a better and more prevalent encoding method is one-hot encoding (Potdar et al., 2017). For each unique category for a given feature, a new column is generated where a binary value represents if the data instance has this category, hence the name “one-hot”. An example of this can be seen in Figure 2.5.



**Figure 2.5:** Example of One-Hot Encoding of Day-of-the-Week.

One-hot encoding therefore preserves the information from the categorical data without introducing the aforementioned issues. At the same time, creating new columns for each categorical value increases memory usage. This may cause the memory footprint of the dataset to balloon given many categorical features with many unique values.



## Data Imbalance

Data imbalance is an issue in ML that occurs when certain categories of data is a significant minority to others. An imbalance in the data can affect the performance of classification models as many algorithms may then neglect minority classes in favor of the majority classes, even though minority classes often are of greater importance (Dube & Verster, 2023).

A model, for example, that aims to detect fraudulent transactions would be greatly affected by data imbalance, as most transactions would not be fraudulent. In the worst case scenario and if the imbalance is large enough, the model may choose to purely output all transactions as regular as this would lead to high accuracy. This causes the model to have a significant bias towards the majority class, and would therefore result in a high number of false predictions (Wei et al., 2013). There are different methods to address this issue.

## Sampling Methods

As mentioned in 2.3.1, equally representing the whole population of a dataset can be a problem. In practical scenarios, datasets rarely cover the entire spectrum of possible outcomes and can lead to models that are not generalized.

A sampling method is a solution to the problem of imbalanced datasets. They work by either removing or generating new data points to improve the balance between classes, with a hope of decreasing the data bias (Brownlee, 2023).

There are numerous techniques available for counteracting imbalances in the dataset, some important methods are mentioned below.

**Oversampling** Oversampling is a technique that produces synthetic data to increase the amount of data points within minority classes in a dataset (Brownlee, 2023). There are many methods for generating such data, with four major methods described in Table 2.1.

Method	Description
SMOTE	Synthetic Minority Over-sampling Technique. This method generates synthetic samples by interpolating between several nearby minority class samples. This aims to improve the decision boundary by making it more general, making the model utilized less susceptible to overfitting (Batista et al., 2004).
Random Oversampling	This method increases the size of minority classes by randomly replicating existing samples in the minority class. This method can lead to overfitting as it simply duplicates existing points (Batista et al., 2004).
ADASYN	Adaptive Synthetic Sampling Approach for Imbalanced Learning. This technique generates synthetic data points by focusing on those regions where the class imbalance is greatest (Batista et al., 2004).
Borderline-SMOTE	A variant of SMOTE where synthetic samples are only generated along the borderline of the minority and majority classes, thus emphasizing the rarer instances (Batista et al., 2004).

**Table 2.1:** Major Methods of Oversampling.

Method	Description
NearMiss	An undersampling technique that selects samples from the majority class based on their distance to the nearest minority class samples. This method aims to equalize class distribution by retaining only those majority class samples that are nearest to minority class samples, thereby fostering a more balanced dataset and potentially improving the decision boundary (Tanimoto et al., 2022).
Random Under-sampling	Reduces the size of the majority class by randomly removing samples. While this method can decrease the problem of an imbalanced dataset, it may also lead to the loss of important information if not carefully implemented, potentially weakening the classifier's performance (Van Hulse et al., 2009).

**Table 2.2:** Major Methods of Undersampling

**Undersampling** Undersampling is a technique that reduces the number of data points within majority classes in a dataset. This approach aims

to balance class distribution by eliminating data, and thereby preventing the models from having bias towards the majority classes. There are several methods for implementing undersampling (Brownlee, 2023), some are outlined in Table 2.2.

### 2.3.2 Validation

Validation, often referred to as evaluation, is how ML performance is quantified. A model will only be as good as the metrics used to validate it, and choosing the wrong metrics may result in the wrong model for the task (Goodfellow et al., 2016).

#### 2.3.2.1 Validation Methods

The hold-out method is an approach to validate the performance of a model using new data the model has previously never seen. The idea is to split the dataset into two disjoint sets – the training set and the testing set. The training set is usually the largest often ranging between 70% - 80% of the total data. After training, validating the model on the test set will mimic how the model would perform in real-world scenarios on future data. While the hold-out method is widely used, according to Roelofs et al., 2019, there has not been much research into the validity and robustness of the method in practical scenarios.

A common strategy for splitting a dataset into a train and test set is simply to randomly shuffle the data. This ensures that the dataset for test and train are completely distinct without overlap.

#### 2.3.2.2 Validation Metrics

Validation metrics in ML are used to understand model performance. As described in 2.3.2.1, this is done on data the model has not learned from. As such, validation metrics help understand how effective the model is at understanding never-seen-before data. The choice of metrics depend on the specific ML algorithm, intricacies of the problem, and desired end-goal.

For certain problems, some metrics may be misleading and lead to a false understanding of model performance (Foster Provost, 1997). For instance, when classifying underperformance in networks, the occurrence of underperformance tend to be rare, for example only 10% of the time. A model that *never* predicts the occurrence of underperformance will still be correct 90% of the time. However, such a model would never correctly identify a case of underperformance, illustrating that accuracy may not be a good metric to understand the performance of such a model.

### Confusion Matrix

The confusion matrix is a visual tool expressing the performance of a classifier, especially useful for problems with skewed datasets (Fawcett, 2006). When a classifier correctly identifies a positive case as positive, it is known as a True Positive (TP). Similarly, a True Negative (TN) is when a classifier correctly identifies a negative case as negative. However, a False Positive (FP) is when a classifier wrongly labels a negative case as positive. Similarly, a False Negative (FN) is when a classifier wrongly labels a positive case as negative. The confusion matrix visualises these concepts.

		Predicted	
		Positive	Negative
Actual	Positive	True positive	False negative
	Negative	False positive	True negative

**Figure 2.6:** The Confusion Matrix.

The confusion matrix can also be defined by the ratio of TPs in relation to FNs, and ratio of FPs to TNs. In mathematical terms:

$$\text{true positive rate} = \frac{TP}{TP + FN}$$

$$\text{false negative rate} = \frac{FN}{TP + FN}$$

$$\text{false positive rate} = \frac{FP}{FP + TN}$$

$$\text{true negative rate} = \frac{TN}{FP + TN}$$

### Accuracy

The accuracy of a ML model is a measure of how often the model predicts correctly. The formula therefore becomes:

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$

Alternatively, using the confusion matrix, accuracy can be calculated as the ratio between TPs plus TNs over all predictions:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

As the example given in the section introduction portrayed, classification problems with heavily imbalanced class distributions are difficult to assess using only the accuracy metric.

### Recall

$$\text{recall} = \frac{TP}{TP + FN}$$

Recall, also known as sensitivity, measures the ability to correctly identify all the relevant instances given to a classifier. Specifically, recall is the proportion of TP predictions out of all actual positives in the data. This means recall becomes the same as the TP rate. A high recall value means that the classifier is effective at correctly identifying positive cases (Powers, 2008).

### Specificity

$$\text{specificity} = \frac{TN}{TN + FP}$$

Specificity is the complement to recall as it measures how many of the negative cases are identified correctly. High specificity means that the classifier is good at identifying negative cases (Powers, 2008).

### Precision

$$\text{precision} = \frac{TP}{TP + FP}$$

Precision measures the accuracy of the positive predictions made by a classifier. In other words, precision measures how many of the predictions labeled positive by the classifier actually belong to the positive class. Therefore, precision expresses how well the model avoids false positives (Powers, 2008).

Similar to the accuracy metric, precision is affected by class imbalance. As the balance shifts, the precision score will in turn change, even if the underlying performance of the classifier remains the same (Batista et al., 2004).

For example, assume a classifier is tasked with detecting a specific disease in individuals. When presented with someone with the disease, the model accurately predicts the presence of the disease 99% of the time (TP rate = 99%). Similarly, given a person free of the disease, the model correctly identifies them as such 99% of the time (TN rate = 99%).

Testing the model on a group of 200 people, 100 with the disease and 100 without, one would expect 99 TPs, 1 FN, 99 TNs and 1 FP. This would result in a precision of 99/100, or 0.99. Using the same model on a group of 101 people, 1 with the disease and 100 without, one would this time expect 1 TP, 0 FN, and the same 99 TNs. Now, the precision drops to 1/2, or 0.5, despite the model staying exactly the same. This example illustrates that when comparing precision values, it is important to understand the impact of the class distribution.

### **F1-Score**

Focusing on maximizing precision can sometimes lead to a decrease in recall, or vice versa. This is known as the precision-recall trade-off – a common phenomenon in many classification models (Florkowski, 2008). To account for this, the F1-Score can be employed. The F1-score expresses this balance by taking the harmonic mean of precision and recall, as seen in the following formula.

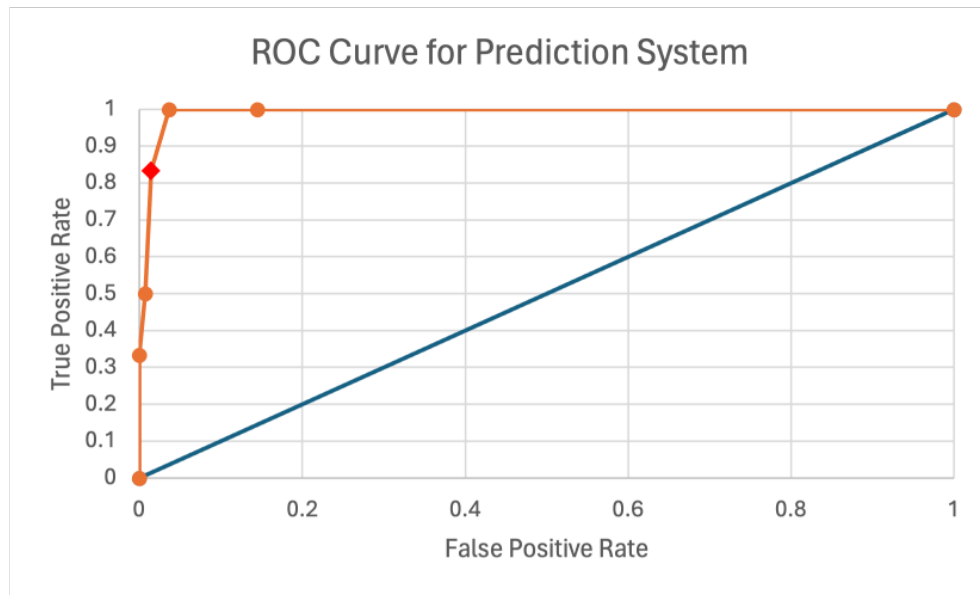
$$\text{F1 score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

(Powers, 2008).

### **Receiver Operating Characteristic**

Up until now, the metrics look at quantifying the error of the predictions. As illustrated by two examples, some of these metrics may be inherently sensitive to class imbalance, which causes difficulties in accurately depicting model performance (Batista et al., 2004). Receiver Operating Characteristic (ROC) graphs do not make any assumptions about class distribution, making it useful when determining the performance of models in problems where this applies (Fawcett, 2006).

ROC curve graphs display the ratio between the TP rate on the y-axis and the FP rate on the x-axis of different configurations of a classification model. By considering a significant number of tuned parameters, the graph provides an overview of model performance. The bottom left point would represent 0 correctly predicted positives, while the top-right point represents 100%. The configuration that meets the acceptable ratio of true-to-false positives can then be chosen; this choice is dependent on the problem being addressed.



**Figure 2.7:** Receiver Operating Characteristic Graph. The Red Dot in The Figure Represents the Model in Figure 2.6.

### Receiver Operating Characteristic Area Under Curve

When comparing models, it is possible to compare their respective ROCs as well. In order to quantify the difference, the Area Under Curve (AUC) metric is utilized. As the name suggests, the AUC finds the area under the ROC curve, ultimately mapping the model's performance to a single value. The value will lie between 0 and 1, where 1 is the best (Powers, 2008).

### Area Under Precision-Recall Curve

Recent research within imbalanced classes have shown that comparison of ROC AUC may lead to an optimistic and inflated understanding of performance (Cook & Ramadas, 2020) (Saito & Rehmsmeier, 2015) (Webb & Ting, 2005). A closely related metric to ROC AUC is the Area Under Precision-Recall Curve (AUPRC). Instead of comparing the area under TP-rate vs. FP-rate, AUPRC quantifies the area under the precision-recall curve, where the precision is on the y-axis and the recall is on the x-axis. Through focusing on one class of a classification problem, AUPRC is therefore representative of the model performance respective to class distributions.

## 2.3.3 ML Model Performance

In ML, performance refers to how well the model achieves its goals and functions. This is quantified using a set of validation metrics, described

in 2.3.2.2. However, looking at performance more abstractly, performance fundamentally is how effective a model can generalize to unseen data (Goodfellow et al., 2016). With this view, two important concepts emerge: over- and underfitting and regularization.

### **Over- and Underfitting**

Broadly speaking, an ML model experiences two types of errors when predicting: training-set error and test-set error. Since the test set contains new input data, the goal of an ML model is to reduce the test set error as much as possible.

The performance of an ML model, according to Goodfellow et al., 2016, can therefore be quantified by its ability to minimize the training error and the gap between the training and test error.

This description of performance underlies the concepts of over- and underfitting. An ML model is underfitted when it is unable to achieve low error on the training set. This characteristic occurs when a ML model struggles to capture the complexity of a dataset. An example of this would be using a linear regression for a dataset created using a quadratic data-generating function.

On the other hand, if the gap between the training error and test error is too large, the model is described as overfitted. Overfitting is the result of a model specializing on the training set rather than the underlying relationships of the data. As such, the model struggles to predict new input data.

To ensure an ML model avoids over- and underfitting, domain knowledge and understanding of the dataset is essential. Moreover, there are a various methods used throughout ML for reducing a model's likelihood to over- and underfit.

### **Regularization**

Regularization are a collection of techniques to modify a ML algorithm with the intent of reducing overfitting. Generally, such techniques trade some training error for added regularity, meaning the algorithm becomes more generalizeable to new data, in turn increasing test accuracy (Goodfellow et al., 2016). This concessions is also called the bias-variance tradeoff. When tuning a model, ensuring regularity is necessary and therefore regularization techniques must be considered ('What Is Regularization?', n.d.).



### 2.3.4 Model Optimization

In this subsection, the concepts and techniques relevant for optimizing a model is explained. Optimization in this case refers to an increase in performance.

#### Hyperparameter Tuning

A hyperparameter is an adjustable value set prior to the learning phase of an ML model. Different hyperparameters influence different aspects of an ML algorithms' behavior. Most ML algorithms expose hyperparameters, allowing for fine-tuning of the algorithm to achieve optimal performance (Goodfellow et al., 2016). This process is called hyperparameter tuning and is a vital part of the training process in ML.

The two most common approaches to hyperparameter tuning is choosing them manually or automatically (Bergstra & Bengio, 2012). Tuning hyperparameters manually is difficult as it requires deep knowledge about the algorithm and the relationship between each hyperparameter and their effect on performance.

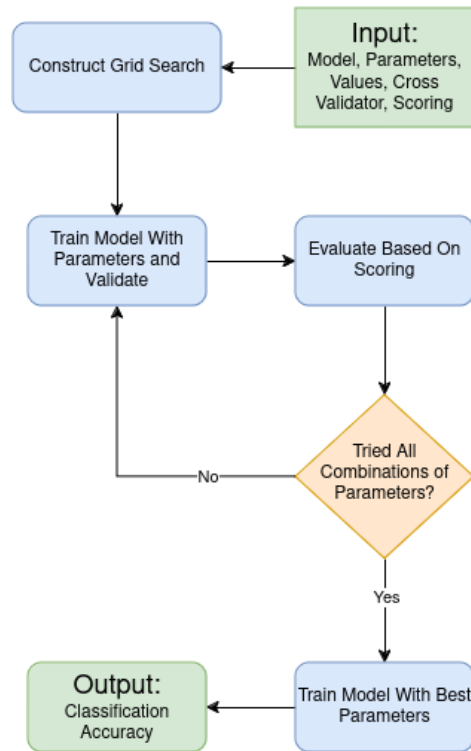
#### Grid-Search

Automatic hyperparameter tuning through a grid-search is therefore a common, but computationally expensive alternative (Goodfellow et al., 2016).

Grid-search works by first choosing a set of values for each parameter to explore. The grid-search proceeds by taking the Cartesian product of each parameter and their values. For each item in the Cartesian product, a model is trained using the corresponding hyperparameter values. Once all variations of the model has been trained, the best model, depending on the metric chosen, is kept (Goodfellow et al., 2016). This method is illustrated in Figure 2.8.

#### Cross Validation

Imagine you need to solve a lot of puzzles, but you are unsure of what puzzle might be presented to you. Therefore, you want to be good at solving any puzzle, not just one. Cross validation is comparable to practicing on many different puzzles and then checking how well you score on average. Now you have an idea of how good you are before you are given a new puzzle. For an ML model, this means splitting the data into different parts, training on some parts and testing on parts it has not seen yet. This helps ensure that a model is generalized and performs well on new data (Berrar, 2018).



**Figure 2.8:** Flowchart Describing The Grid-Search Process. Inspired by Uddin et al., 2021.

**K-Fold** K-Fold cross validation is a method in which the dataset is randomly split into  $K$  equal sized distinct subsets. In this way, no two subsets overlap. Subsequently, the ML model is trained on  $K - 1$  subsets of data, with the last being used as a validation set. This process is repeated until all  $K$  subsets have been used as the validation set (Berrar, 2018).

**Stratified K-Fold** Stratified K-Fold cross validation works much in the same way as regular K-Fold, although it has a slight difference. By stratifying the subsets, each subset is ensured to have the original distribution of each class, ensuring a more accurate validation. This is especially important when working with imbalanced datasets (Neyman, 1992).

### 2.3.5 Explainability and Interpretability

Within the realm of AI research, the definitions of interpretability and explainability are usually domain-specific, yet an important distinction (Marcinkevičs & Vogt, 2023). Although some research papers do not differentiate between the two concepts, in this paper, interpretability will describe

the conceptual understanding of the model architecture, often referred to as algorithmic transparency. On the other hand, explainability will refer to the degree of understanding how a model makes its predictions. Understanding, in this sense, is the ability to accurately describe how a given input will impact the output (i.e model prediction).

As AI systems continue to be deployed into critical systems, the concept of explainability (Dwivedi et al., 2023) and interpretability (Rudin, 2019) becomes increasingly important. Given this importance, a new area of research called Explainable AI (XAI) has emerged which aims to “provide a suite of machine learning techniques that enable human users to understand, appropriately trust, and produce more explainable models” (Dwivedi et al., 2023).

### **Intrinsic Interpretability**

As the name suggests, with intrinsic interpretability, models are pre-disposed to being understandable purely due to their construction. Models with intrinsic interpretability are generally referred to as glass box models. With this type of interpretability, the focus is on the models themselves being understandable rather than just the results (Marcinkevičs & Vogt, 2023).

### **Post-hoc Explainability**

To capture more complex relationships of data, there is usually the necessity for more powerful ML models. Following this, there is a general negative correlation between model complexity and explainability (Ribeiro et al., 2016). In other words, as an ML model becomes more complex its innerworkings become less understandable to humans.

Post-hoc explainability is therefore an attempt to understand what an ML model values after having been trained. When discussing such techniques, there are broadly speaking two categories: model-specific and model-agnostic techniques. Model-specific techniques are methods created with a focus on a type of model, while model-agnostic techniques are applicable to a larger variation in model type (Molnar, 2022).

### **Impurity-based Feature Importance**

Mean Decrease in Impurity (MDI) is a model-specific, post-hoc explainability technique for tree-based models. For a given feature, the feature importance is determined through totaling the decrease of the chosen impurity metric (Gini index, variance of output, etc.) for the internal nodes of a tree and averaging it amongst all trees in the model (Louppe et al., 2013).

### Permute-and-Predict Methods

With permutation importance, the goal is to determine the influence of specific features through deliberate alterations and recording the changes. Formally, Permute-and-Predict (PaP) is performed through calculating the difference in the loss function for a permuted entry vs. the original entry. The changes in error are calculated using predictions from the trained model. A permutation is a random selection of a value from the sample space of a given column. Therefore, a single data entry, or row, will swap out a specific feature's value with another entry's value. This process is repeated N-number of times for a given feature and the sum of differences equates to the variable's importance (Hooker et al., 2021).

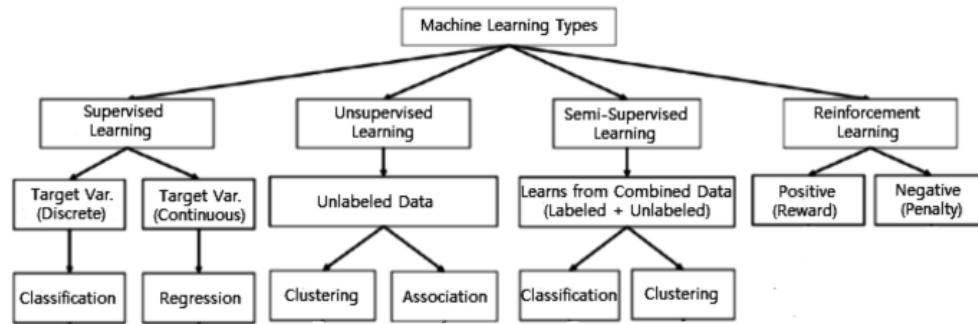
When utilizing PaP methods, it is important to understand the potential downfalls. In Hooker et al., 2021, the authors discuss the potential misrepresentation of feature importance when features are correlated with one another.

### SHAP Values

SHapley Additive exPlanations (SHAP) values are both global and local, model-agnostic post-hoc explainability methods (S. Lundberg & Lee, 2017). Global, in this context, refers to the scope of the explanation being the whole model rather than instance-based (local). In S. Lundberg and Lee, 2017, the authors utilize the Shapley regression values in order to create the SHAP values. The Shapley regression value is found through calculating the difference in model performance with and without the feature in the dataset.

### 2.3.6 Supervised Machine Learning Algorithms

ML systems can be divided into four main categories: *reinforcement learning*, *supervised learning*, *unsupervised learning* and *semi-supervised learning* (Mohammed et al., 2016).



**Figure 2.9:** Diagram Depicting Various Machine Learning Categories Taken from Sarker, 2021.

The choice of ML category for a given problem depend mostly on two variables: The form of the available data, and the end-goal of the model (Sarker, 2021). For instance, given a labelled dataset, supervised learning tends to be preferred as such models utilizes these labels in an attempt to learn the intrinsic structure in the data.

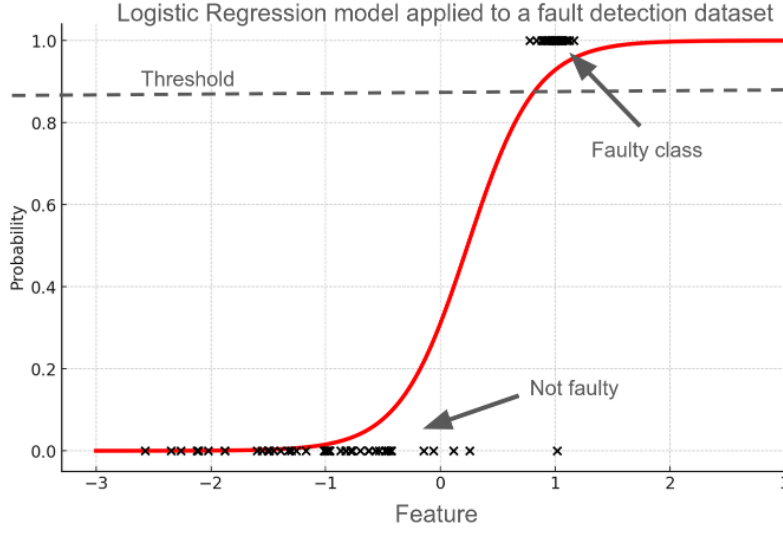
### Decision Tree

A decision tree is an ML model where the result of fitting the model is a hierarchy of conditional statements. When a prediction is made, the model takes in a data point and traverses down the tree. At different branch nodes, the data point is evaluated against the conditional statements until a leaf node is met, resulting in the prediction. Due to the simplicity in going from input data to an outputted prediction, decision trees are considered highly explainable (Khalil et al., 2021). Additionally, decision trees have the benefit of grasping non-linear relationships. Some of the downfalls, however, are the insufficient capacity to capture highly complex relationships, higher risk of overfitting, and poor performance with imbalanced datasets (Hastie et al., 2009).

### Logistic Regression

Logistic regression is a relatively simple binary classifier. It is widely used in medical research for predicting the probability of the presence or absence of a disease (Kirkwood & Sterne, 2010). Logistic regression is conceptually similar to linear regression, except it attempts to predict whether something is true or false instead of something continuous.

Logistic regression fits an "S" shaped *logistic function* to the data as can be seen in Figure 2.10. For each data point, a probability that the data point fits into either category can be read from the curve given the input feature.



**Figure 2.10:** Logistic Regression Applied to an Example Dataset Using an Arbitrary Threshold Value.

For this reason, the threshold or tolerance for categorizing a data point into either the positive or negative class becomes a modifiable parameter that can be tuned for the specific task.

Logistic regression can also accept multiple features as input. Given multiple features, the formula is extended to accommodate each feature. The result is a curve in multiple dimensions. The formula for multiple logistic regression is per LaValley, 2008:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

Here,  $p$  is the probability of the outcome,  $\beta_0, \beta_1, \dots, \beta_n$  are the coefficients, and  $X_1, X_2, \dots, X_n$  are the input features.

### Random Forest

The Random Forest (RF) model is an enhanced form of the decision tree, attempting to bring along the advantages while leaving the disadvantages. RF utilizes a method called bootstrap aggregating, further called bagging, which involved creating multiple decision trees by training each tree on a random subset of data. By doing this, the model reduces the risks of overfitting by introducing diversity within the trees in the forest (Altman & Krzywinski, 2017).

This can be represented mathematically as the following:

Let  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$  represent the training dataset with  $N$

samples, where  $x_i$  denotes the feature space and  $y_i$  denotes the target. Bagging involves repeatedly sampling  $N$  examples from  $D$  with replacement to create  $B$  bootstrap samples, denoted as  $D^{(b)}$ , where  $b = 1, 2, \dots, B$ .

Each decision tree in the RF is trained on a different bootstrap sample  $D^{(b)}$ . During tree construction, at each node, a random subset of features is considered for splitting, adding further randomness to the model. This process helps to decorrelate the trees and improve the overall generalization performance of the RF (Zhan et al., 2021).

As the RF model employs multiple decision trees, it falls under the category of ensemble learning. Ensemble learning is the technique of utilizing and combining various models to form a singular, but better model. With the increased complexity of multiple decision trees, the RF model loses some of the explainability associated with decision trees.

### Boosting Algorithms

Boosting algorithms is a subset of supervised ML algorithms that is based on converting weak learners, a model slightly better than a random guess, into strong learners. This is done by training models, where each model tries to correct the errors of the previous models. The final model is then a weighted sum of all the individual models.

The general form of a boosting algorithm can be expressed as:

$$F(x) = \sum_{t=1}^T \alpha_t f_t(x)$$

where  $f_t(x)$  is the  $t$ -th weak learner,  $\alpha_t$  is the weight assigned to the  $t$ -th learner, and  $T$  is the total number of learners.

**CatBoost** Categorical Boosting (CatBoost) is an open-source gradient boosting library for handling categorical data. It works by transforming categorical values into numerical values during training. CatBoost is an ordered boosting algorithm and reduces overfitting through permutation-driven independent training on random data subsets (Dorogush et al., 2017).

**XGBoost** Extreme Gradient Boosting (XGBoost) is an open-source gradient boosting framework that is effective with large datasets. It uses a regularized model to control overfitting and constructs an ensemble of decision trees sequentially, where each tree corrects the errors of the previous ones (T. Chen & Guestrin, 2016).

**LightGBM** Light Gradient Boosting Machine (LightGBM) is a gradient boosting framework for efficient and distributed training, particularly on large datasets. It uses gradient based one-sided sampling and feature bundling to reduce computation costs. LightGBM grows trees leaf-wise rather than level-wise, which can lead to better error reduction (Ke et al., 2017).

**AdaBoost** Adaptive Boosting (AdaBoost) is a gradient boosting method that sequentially trains weak classifiers, focusing on examples misclassified by previous classifiers. The final prediction is a weighted sum of the individual classifiers, with weights adjusted to minimize the training error.

The training error  $E_t$  is minimized at each iteration:

$$E_t = \sum_i E [F_{t-1}(x_i) + \alpha_t h(x_i)]$$

where  $F_{t-1}(x)$  is the boosted classifier from the previous stage, and  $f_t(x) = \alpha_t h(x)$  is the current weak learner (Freund & Schapire, 1997).

### Support Vector Classification

Support Vector Classification (SVC) is a supervised algorithm for classification that bases itself on kernel functions. The goal of SVC is to find the hyperplane that best separates different classes in the feature space. The hyperplane is chosen to maximize the margin between the classes, where the margin is defined as the distance between the hyperplane and the nearest data point from either class. The data points on the margins are referred to as support vectors and what define the hyperplane. SVCs are particularly effective in high-dimensional spaces and when the classes are not linearly separable (Cristianini & Ricci, 2008).

### 2.3.7 Deep Learning

Deep learning is a class of ML algorithms that are inspired by the structure and function of the human brain. Deep learning methods have, in a very short time, drastically enhanced the state-of-the-art in genomics, computer vision, speech recognition, and more (LeCun et al., 2015).

While conventional ML techniques require careful engineering to go from raw data to model features, deep learning techniques are often able to automatically learn patterns from raw data. This is because deep learning models have multiple layers that represent and understand the data at different levels of abstraction. This idea has generally proved to be very effective at learning complex functions from input to output, even when the input dimensionality is high (LeCun et al., 2015).



Despite its effectiveness, the internals of deep learning models tend to be particularly complex. A consequence is that the degree of explainability tends to be low, especially as model complexity increases (Dwivedi et al., 2023). Another sticking point, making deep learning unsuitable for many tasks is the high compute power associated with using such models for both training and deploying (LeCun et al., 2015).



## **3. Method**

The method chapter of this thesis will step by step guide the reader through how the research project was performed, aiming to ensure the reproducibility of the results. All parts of the project will be described, from how the project was planned to how these plans were followed through and how any setbacks were handled – looking at both the administrative and practical aspects. Understanding how the results, presented in Chapter 4, were gathered will be critical when reading the discussion in Chapter 5.

### **3.1 Planning and Deciding the Research**

This section will describe how and why the problem was chosen. In addition, the process for narrowing the scope to a manageable yet relevant problem statement for the thesis will also be detailed.

#### **Assignment Scoping**

The problem description along with the accompanying dataset gave way for several areas worthy of further exploration. However, focusing on every possible aspect would not be possible. Consequently, identifying the most interesting areas of the problem to focus on became the initial task. During this process, specific criteria were applied to ensure the chosen research area maintained industry and academic importance. At the same time, the chosen area must remain feasible with relation to time limitations, available dataset, and the group's level of competence. Table 3.1 outlines these criteria.

Criteria	Description
Academic relevance	To ensure this, a literature review was performed, where interesting and relevant discoveries along with unanswered questions were noted. Later, each proposed research question had to be anchored in interesting discoveries or unanswered questions within the research area.
Industry relevance	As will be discussed in 3.2.2, an interview with Telenor was held to ensure the proposed research questions were of interest to telecommunication operators.
Feasibility	Firstly, an initial exploratory analysis, which will be discussed in 3.4.2, of the dataset was performed to understand what was possible given the available data. Any proposed research question had to fit within the possibilities of the available data. Furthermore, meetings with both supervisors were held where the feasibility of the proposed research questions were discussed. Finally, the team made attempts to estimate time and difficulty for each proposed research question.
Novelty	Although not a strict requirement, novelty was positively factored in when deciding on which direction to narrow the scope of the thesis.
Ethics	Any proposed research question had to align with the team's ethical principles.
Motivation	The personal motivation and interest in the proposed research questions was also factored in.

**Table 3.1:** The Criteria Employed When Deciding What Direction to Take the Thesis.

## Planning

Early planning was done in a coarse-grained manner using GANTT and Activity-On-Arrow diagrams. The purpose of these plans were to provide a high-level overview of the project timeline and dependencies without going into fine-grained detail. The decision to not perform any fine-grained planning is based on uncertainties in estimating time for certain research aspects of the thesis. Instead of planning with time estimates, a comprehensive flow chart diagram of the development cycle was made, seen in Figure 3.5.

## Team Roles and Work Distribution

As a measure to ensure high quality work, each group member was the given certain roles and responsibilities. One such role was meeting transcriber whose responsibility was to make sure meeting proceedings were written. Roles were static and did not change during the project.

Further, to ensure fair work distribution when developing the technical aspects of the project, the project management tool Jira<sup>1</sup> was utilized. This ensured all tasks were written in a centralized place, and team members could keep track of progress. In conjunction, with the timesheets detailing the hours worked by each team member, assurance of fairness in workload could be achieved. Usage of Jira and the timesheets is further detailed in 3.3.4.

## 3.2 Research Approach

The purpose of this section is to provide the basis for the choices taken during the research process. Only through a clear and understandable process is it possible to develop quality research. One of the most important aspects of this understanding is the ability to reproduce results and to evaluate the underlying strengths and weaknesses of the research. Therefore, the first part of this section will go through an overview of the scientific method. Thereafter, the more focused research design choices are reviewed.

### 3.2.1 Scientific Method

The scientific method was used as a baseline as it is essential when aiming for objective research of high-quality. Generally, the scientific method consists of an information gathering stage, problem-formulation and/or hypothesizing, implementation and experimentation, and finally validation and reflection.

### 3.2.2 Research Design

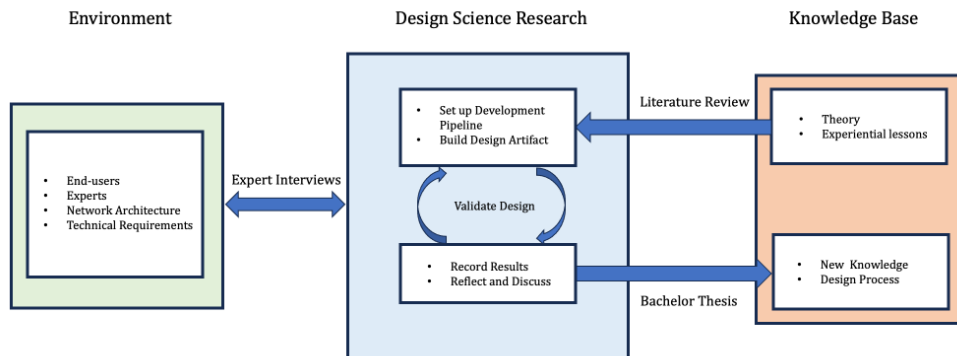
With a basis in the scientific method, the specific research methodology chosen for this thesis is the Design Science Research (DSR) model.

DSR is a paradigm and framework that bridges developmental and theoretical research methodologies (Wrålsén & Berntsen, n.d.). Furthermore, with DSR, there is a focus on both the artifact created (the product) as well as the context (its environment) (Wieringa, 2014). For this project, the

---

<sup>1</sup><https://www.atlassian.com/software/jira>

artifact will be the predictive system for underperformance. Consequently, the purpose of the research is to predict and understand a problem rather than uncover fundamental answers. It is therefore necessary to consider the practical details surrounding the relevance of the problem. Hence, the context is how such a system can be integrated down the line with network operators.



**Figure 3.1:** Research Method Model. This Model is Inspired by Hevner (2007, Fig. 1, p. 3)

As discussed by Hevner, 2007, DSR can commonly be described by three cycles: the relevance cycle, the design cycle, and the rigor cycle. For larger projects, focus on all three cycles is mandatory. However, due to time limitations, this thesis prioritizes work within the design cycle, characterized by the center component in Figure 3.1. The research specific details surrounding the design cycle is discussed in Section 3.4. Furthermore, the team views it necessary to supplement the design cycle with selected aspects from the other cycles.

### Information Gathering and Literature Review

To map the progress and limitations surrounding the problem, a literature review was undertaken at the start of the project. The problem exists in the intersection between two major fields: ML and SON in cellular networks. Therefore, a literature review was necessary for the team to gather theoretical information about the intersection of the two domains. In the model proposed by Hevner, 2007, this process is an interaction between the knowledge base and the DSR.

One of the deciding factors for valid research at this stage is determining the credibility of sources. During the literature review, the team solely utilized sources grounded in the scientific method and which were peer-reviewed. To ensure these qualities, research papers were found through the credible, scientific literature search engine Google Scholar. Additionally, the team used academic textbooks that are referenced in other credible sources or from curriculum in respected universities.

During the literature review, emphasis was placed on not forming opinions while reading through the literature, but instead attempting to form a broad understanding of all sides of the problem. To do so, minimizing cognitive biases such as anchoring and confirmation bias is important.

Anchoring bias is the tendency to believe and trust the first source of information or idea for a given topic (Behimehr & Jamali, 2020). In this research project, the team reduced anchoring bias through constant focus on minimizing this bias through gathering multiple sources on a topic before drawing conclusions.

Confirmation bias occurs when a person seeks information that supports their belief or understanding (Mohanani et al., 2020). This type of bias removes objectivity from the information gathering stage and can have serious consequences for the validity of the research. To reduce this bias, the team held frequent discussions surrounding critical information and all members were responsible for quality assuring other member's understandings.

Although acknowledgement and identification of bias is important, it does not necessarily reduce it (Heuer, 1999) (Aczel et al., 2015). However, one way of counteracting a recognized bias may be through exercising contradiction (Mohanani et al., 2020)(Heuer, 1999)(Aczel et al., 2015). This contradiction can be done through imagining an opposite scenario or counterfactual evidence. During discussions, the team therefore worked to argue for and against the common understanding.

### **Interview with Experts**

To ground the research with the context, the team held an interview with one of Norway's largest telecommunication companies, Telenor. The goal of the interview was to better understand the technical details surrounding integration of solutions, current goals of operators, and the importance of explainability. It is important to note that the interview purely serves the purpose of providing supplemental empirical data to the research.

The interview was a semi-structured interview. This type of interview is

characterized by having a list of pre-made questions from which discussion arises. This type of interview format was chosen due to its flexibility in exploring topics of interest that arise during the interview. The team purposefully left questions open-ended. The reason for this decision was to reduce the influence the team had on the answers and to allow the interviewees to emphasize the topics they felt were important.

The interview was held digitally over Teams on the 17th of April, 2024. All usage of information from the interview in this thesis was approved from the interviewees.

### Evaluating the Design

In Table 3.2, the most significant strengths and weaknesses of the chosen research design are outlined.

Strengths	Weaknesses
Promotes higher quality research through focus on valid and credible data	Time-consuming.
Aims at minimizing effects of bias.	All bias cannot be removed.
Encourages being critical and reflective of own and other members' research.	Can be demotivating to constantly have a devil's advocate.
Motivates the research through highlighting its context	

**Table 3.2:** Strengths and Weaknesses of Chosen Research Design.

## 3.3 Development Approach

A standardized approach to the development process is necessary to ensure both consistency and high-quality. In this section, the standardized practices, rituals, and technologies utilized in the project will be outlined.

### 3.3.1 Choice of Technologies

#### Programming Language and Third-Party libraries

Python was chosen as the language-of-choice for all programming matters of the thesis as it stands as the de-facto standard language for ML, statistics and data processing. An important reason for Python's prevalence, and also the teams decision to use it, is its rich ecosystem of third-party libraries. Table 3.3 shows a collection of valuable third-party Python libraries that were used to develop the ML pipeline. While Python typically



is relatively slow compared to other languages, most third-party Python libraries are written in the C language and have been highly optimized, making them very fast and suitable for ML on large datasets.

Library	Purpose
Matplotlib (Hunter, 2007)	Data visualization
Pandas	Data manipulation and analysis
imbalanced-learn (Lemaitre et al., 2016)	Sampling methods
scikit-learn (Pedregosa et al., 2011)	Basis for various ML implementations
SHAP	Post-hoc Explainability Techniques
PyTest	Unit testing
XGBoost	XGBoost implementation
LightGBM	LightGBM implementation
CatBoost	CatBoost implementation

**Table 3.3:** Collection of Most Valuable Third-Party Python Libraries.

Another reason for choosing Python was its familiarity to all group members. Additionally, its ease of use and flexibility made it a good fit.

### High Performance Computing

The sheer amount of data meant training and testing the ML models on regular laptops was not possible. This is because the implementations of the various ML algorithms need the entire dataset available in RAM during training. Additionally, depending on the algorithm and data size, the time required to train the models could exceed the time available for the project.

To solve all these issues, the team requested access to NTNU's High Performance Computing (HPC) facilities, IDUN. Once access was granted, running the models requires creating a job through Slurm<sup>2</sup>. Figure 3.2 illustrates the standardized shell script developed for this purpose.

For deployment, an important aspect for each ML model is the time needed to fully retrain and to make a prediction. To make such comparisons between models, a necessary prerequisite is standardizing the hardware across all benchmarks. Table 3.4 outlines the hardware used across all tests on IDUN. This is also reflected in the Slurm script in 3.2.

Type	Processor	Cores	RAM (GB)	Experiments
Dell C6520	Intel Xeon Gold 6348 (CPU)	32	128	Model Comparison, Granularity
Dell C6420	Intel Xeon Gold 6242 (CPU)	32	128	Centralized vs Distributed

**Table 3.4:** Hardware Specification Used to Train and Test Models.

<sup>2</sup><https://slurm.schedmd.com/overview.html>

```
#!/bin/sh

#SBATCH -N 1          # Allocate 1 nodes for the job
#SBATCH -c 20         # Idun 20 cores
#SBATCH -t 00:01:00   # Upper time limit for the job (DD-HH:MM:SS)
#SBATCH --partition=CPUQ
#SBATCH --mem=32G      # 32 gigabytes memory
#SBATCH --job-name="Job name"
#SBATCH --output=model-output.txt
WORKDIR=${SLURM_SUBMIT_DIR}
cd ${WORKDIR}

#module load intel/2023b
module load Anaconda3/2022.10

if [ $# -eq 0 ]; then
    echo "No file arguments"
    exit 1
fi

file_path=$1

python $file_path
```

**Figure 3.2:** Slurm Script Used For Running Batch Jobs on IDUN.

### 3.3.2 Ensuring Code Quality

A major portion of the project is experimenting with statistical techniques and ML algorithms through programming. Therefore, it is essential to work in an organized manner.

To coordinate changes to the codebase, the team used the central version control software Git<sup>3</sup> and the platform GitHub<sup>4</sup>. With Git, changes to the code are reflected in commits. The Angular commit convention<sup>5</sup>, was used to systematize these changes. The convention specifies how the commit message should be formatted, making it easier to understand changes.

Whenever changes are made, the team creates a new branch to reflect the change. By doing so, the team always maintains a stable version of the progress. Furthermore, once the new branch is evaluated as stable by the owner, the other team members were required to review and accept the changes before merging. This step of reviewal was multi-purposed. A review generates discussion surrounding ideas and allows other members to stay up-to-date with progress.

When it comes to programming, the group applied best practices regard-

<sup>3</sup><https://git-scm.com/>

<sup>4</sup><https://github.com/>

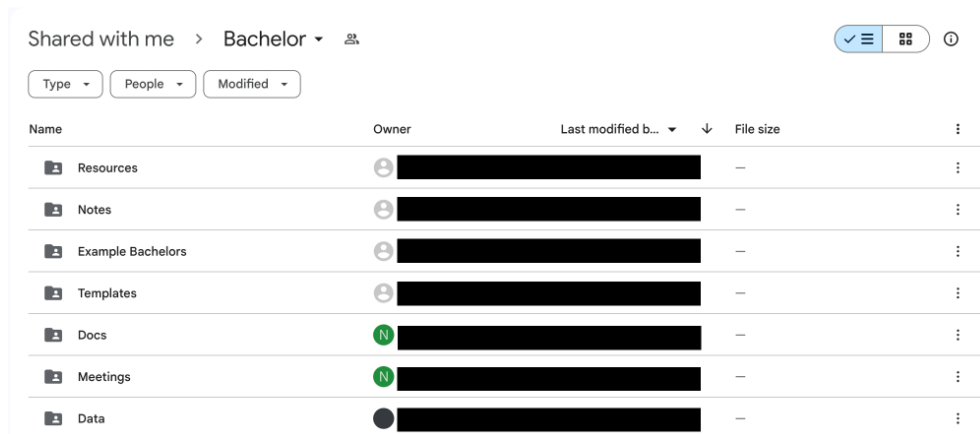
<sup>5</sup><https://www.conventionalcommits.org/en/v1.0.0-beta.4/>

ing readability, efficiency, and robustness of the code. For readability, the group followed the official syntax conventions for Python as described in the PEP8 guidelines<sup>6</sup>.

For the most critical sections of the codebase, unit tests were written to ensure correctness. The section deemed critical were the filtering and processing code for the dataset. A potential error here would have significant ramifications for the correctness and efficacy of the model.

### 3.3.3 Documentation

Recording ideas and progress is a vital part of research. To streamline this stage, the group used Google Drive as the main storage location. In Google Drive, the group created a hierarchy of different folders to organize documentation.



Name	Owner	Last modified b...	File size
Resources	[Person Icon]	[Redacted]	—
Notes	[Person Icon]	[Redacted]	—
Example Bachelors	[Person Icon]	[Redacted]	—
Templates	[Person Icon]	[Redacted]	—
Docs	[Person Icon]	[Redacted]	—
Meetings	[Person Icon]	[Redacted]	—
Data	[Person Icon]	[Redacted]	—

**Figure 3.3:** Structure of Google Drive.

Recording the experiments results was done through a script that wrote out the performance metrics along with other information of interest. This data was saved in a CSV format, ensuring results could be analyzed and compared at any time.

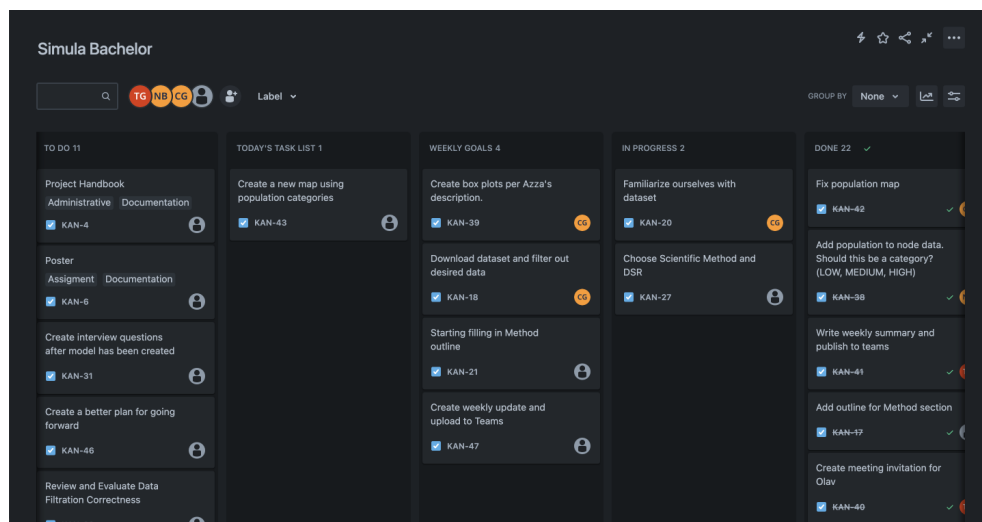
### 3.3.4 Progress Reports

Reflection is an important part of the development process. The team has therefore introduced weekly summaries into the work process. At the start

<sup>6</sup><https://peps.python.org/pep-0008/>

of the week, the team determines the milestones and goals for that week, based on recent progress and the overall plan outlined in the GANTT-diagram. At the end of the week, the team gathers to discuss the progress made in reference to the goal and to evaluate the path forward. This way of working is reminiscent of the Sprint retrospective event found in Agile methodologies. Additionally, by periodically archiving the progress, the team has the opportunity to look back at previous work.

Similarly, for the purpose of maintaining an overview of the work, the team utilizes a Kanban board on the platform Jira.



**Figure 3.4:** The Team's Kanban Board on Jira

As seen in Figure 3.4, the group distinguishes tasks based on time and priority. Each task can be assigned to a group member, or worked on by everyone. By maintaining an up-to-date Jira, team members were able to efficiently coordinate work.

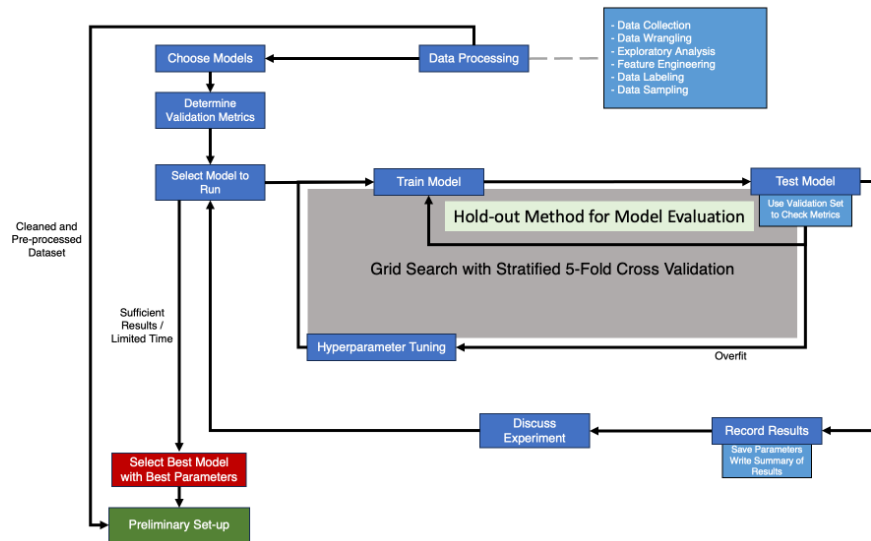
### 3.3.5 Standardized Communication

To best utilize the supervisors as a resource, the team established a routine of biweekly meetings with the supervisors. During these meetings, the group's transcriber made sure to write notes of the key points discussed. At the end of each meeting, the group then made sure to go over the notes with the respective supervisor to ensure their correctness. Later, these notes were refined and integrated into a meeting report. This report encompassed all relevant information about the meeting such as the agenda, time, date, and attendees present.

Microsoft Teams<sup>7</sup> was adopted as a standardized communication channel with the supervisors. Here, documents of interest such as the weekly summaries and other artifacts were shared and kept up-to-date as progress continued.

### 3.4 Preliminary Setup

In this section, the initial experimentation and development that lead up to the preliminary setup will be described. All subsequent experiments will build on this foundation. Firstly, the initial data exploration will be described. Next, the filtration process will be detailed along with the process of transforming the raw data into a procured dataset. Subsequently, the methods used to find the best performing ML model for the dataset will be described. A coarse-grained overview of the general pipeline for the development process is depicted in Figure 3.5.



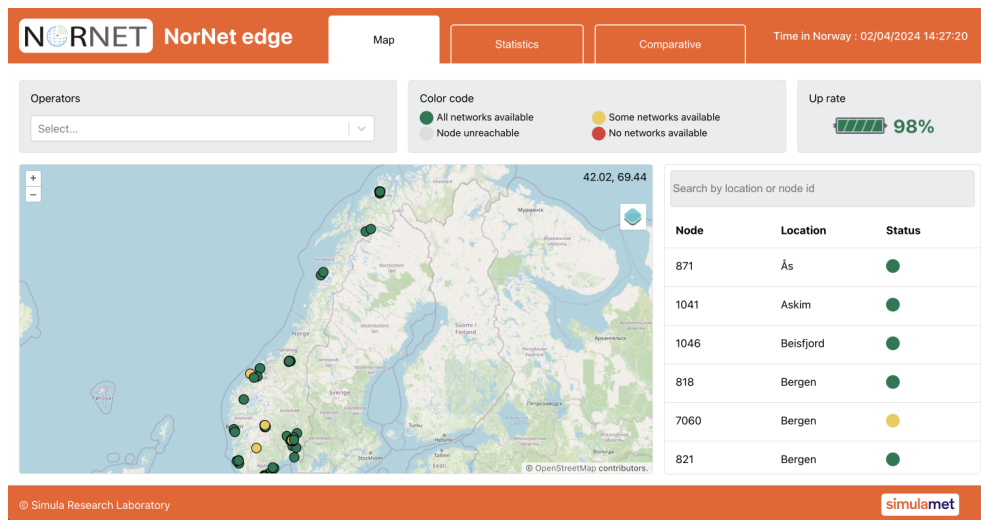
**Figure 3.5:** The Preliminary Development for The ML Pipeline.

Although Figure 3.5 describes each step in the process, certain modifications had to be made due to unforeseen elements that arose during development. In the following sections, each component from the diagram will be further described while the aforementioned unforeseen elements will be explained in 5.6.3.

<sup>7</sup><https://www.microsoft.com/en-us/microsoft-teams/group-chat-software>

### 3.4.1 Data Collection

The first step in the development process was collecting a sufficient data-set. Since this research paper focuses on end-user data, SimulaMet provided the team with raw data collected from the measurement devices from the NNE platform. NNE is an infrastructure of over 400 geographically-spread capturing devices in Norway working on measuring the network state of fixed and wireless mobile networks (Kvalbein et al., 2014).



**Figure 3.6:** Map Visualization of NorNet from RobusteNett Website.

These NNE capturing devices will hereon be referred to as *nodes* in the NNE infrastructure. The nodes collect end-to-end measurements through the internet via commercial mobile subscriptions. Every second, a 20-byte UDP packet is sent over all available connections (Ahmed et al., 2021). These measurements give data for RTT, jitter, and packet loss. Additionally, each node collects passive metrics from the network such as RSSI, RSRP and RSRQ. This data is automatically gathered when a passive metric changes value. For each data point, a timestamp of when the measurement was taken is included.

These NNE nodes gather the data from all three major Norwegian mobile providers. Through standardized hardware and software amongst all the nodes, the end-to-end experience of the cellular networks can be measured in isolation. The hardware of the nodes in the NNE infrastructure was therefore carefully chosen to allow for high comparability to the average user; this is exemplified in the use of off-the-shelf hardware such as the Huawei E353-u2 3G USB modem for collecting 3G measurements (Kvalbein et al., 2014) (Ahmed et al., 2021). This thesis however only looks at

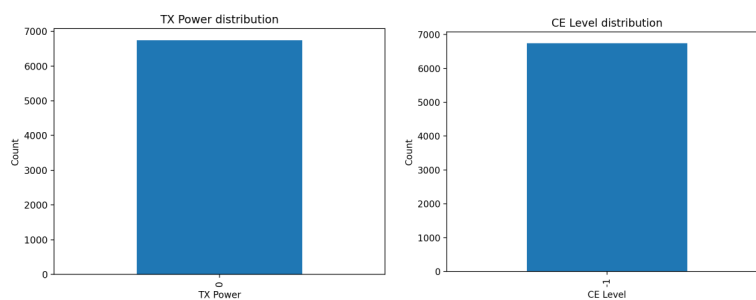
the 5G data collected from the NNE platform.

Furthermore, to narrow down the scope to a manageable yet meaningful size, only eleven out of the 400 nodes will be considered in this thesis. The eleven chosen nodes are all located in the Oslo Metropolitan Area. Further, three months of measurements were considered. Finally, all data from the eleven chosen nodes are from the same telecommunication operator to ensure consistency. To adhere to privacy desires, the team made sure to preserve the privacy of the chosen telecommunication operator.

While Colpitts and Petersen, 2023 found only a minimal improvement in model performance when including exogenous information using a rural dataset, it remains interesting whether adding exogenous contextual information, such as population, would improve model performance for an urban dataset. As such, to supplement the dataset with exogenous contextual information, population data was utilized. The population dataset from SSB, 2023 had a granularity of  $1 \text{ km}^2$  and was retrieved from GeoData Norge<sup>8</sup>.

### 3.4.2 Initial Exploratory Analysis

When the data was made available, one of the first things done was to simply see what was there. To do this in a structured manner that was understandable, plots were utilized to both categorize the values of different metrics and to see trends. Through this, it was found that some values were always constant. Some of these plots are shown in Figure 3.7.



(a) Distribution of TX Power Values in Metadata Events. (b) Distribution of CE Values in Metadata Events.

**Figure 3.7:** Plots of Constant Values Found Initial Exploratory Analysis.

<sup>8</sup><https://www.geonorge.no/>

### 3.4.3 Initial Dataset Selection

From the data exploration mentioned in 3.4.2, we can see in Figure 3.7 that the metrics TX Power and CE level only have one value. This meant that they could then be discarded as they would only bloat the dataset without providing any value. In addition to the exploration, the project supervisor at SimulaMet, PhD. Azza Ahmed, provided a detailed description of the dataset, highlighting the measurements they understood to be relevant for the thesis. By combining, the exploratory findings with the dataset description, the final choice of network metrics to use was chosen. See Table 3.5.

Metric	Description
RTT	Round-Trip-Time of the 20-byte UDP packet sent every second. The time is measured from when the first bit of the packet and stops when the final bit is received.
RSSI	Received Signal Strength Indication. Passive measurement received infrequently.
RSRQ	Reference Signal Received Quality. Passive measurement received infrequently.
RSRP	Reference Signal Received Power. Passive measurement received infrequently.

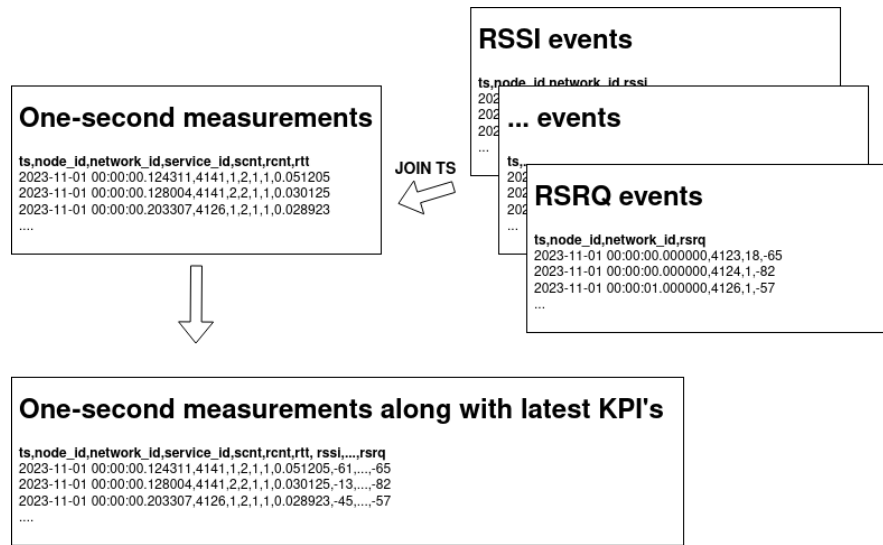
**Table 3.5:** Metrics Chosen After Initial Dataset Selection.

### 3.4.4 Merging the Dataset

The different parts of the dataset had different granularities of measurements. For instance, RTT was measured every second, while metadata events were received sporadically. To connect these measurements together into a large, single dataset, merging was required.

This merging was done based on the timestamp associated with each measurement. The approach taken was to connect the most recent metadata event for each KPI to the past one-second RTT measurement. Figure 3.8 visualizes this. The result is one unified dataset containing the one-second measurements with the most recent KPI values attached.





**Figure 3.8:** How Passive Metadata Events Were Merged into the One-Second Measurements to Form a Unified Dataset.

### 3.4.5 Data Wrangling

Even though the dataset was collected and subsequently merged, it remained in a raw format unsuitable for practical use. Therefore, the process of transforming the raw data into a usable format was a key aspect and the foundation for further ML development.

Firstly, only one-second measurement data that contained both a sent and a corresponding received packet were considered for further analysis. This is because if either one is missing (likely due to a malfunction in the data capturing device), the data is incomplete and will only add noise to the model. Secondly, an NNE node may experience downtime during the three-month period, resulting in no measurements being taken. Consequently, all periods of downtime were removed.

Finally, some data points contained invalid data. These were also likely due to malfunctions in the data capturing device. For instance, RSSI can be defined in the interval between -100 dBm and -6 dBm. A value outside of this range should not be possible and such events were discarded. Table 3.6 shows the valid intervals for the data points where malfunctions may occur. The values for this table are based on the conservative values for these metrics found in Ahmed et al., 2021.

Feature	Min value	Max value
RSSI	-100 dBm	-6 dBm
RSRP	-140 dBm	-44 dBm
RSRQ	-20 dBm	-3 dBm

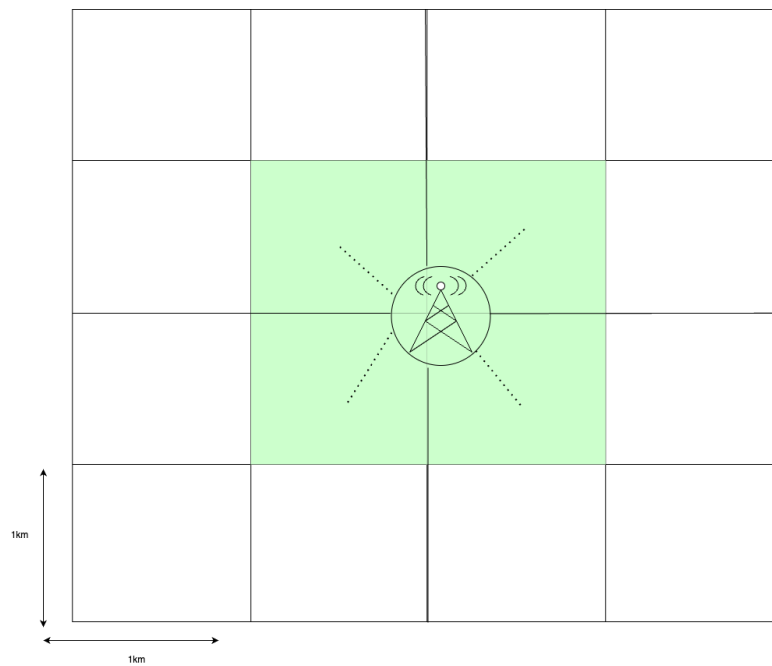
**Table 3.6:** Valid Feature Intervals.

In addition to removing entries with bad KPI-values, it was decided to remove RTT-values that showed significant deviations from the mean. Again, such values would only add noise and negatively contribute to model performance. To remove these, Tukey's IQR method, as described in 2.2.2, was employed. This ended up removing  $\sim 4.5\%$  of the dataset.

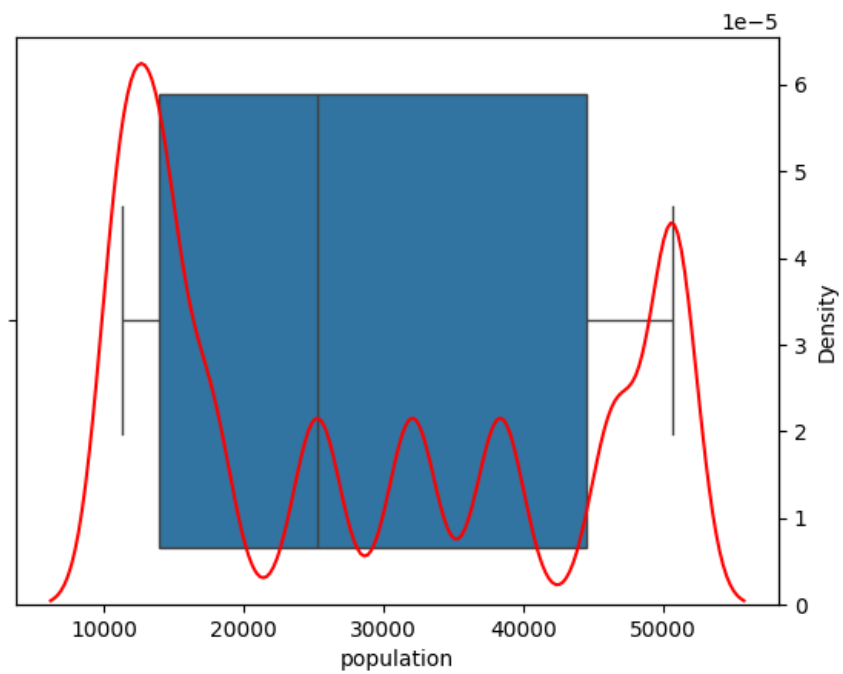
Finally, certain features must be categorized before they can be understood by an ML model. This was done using one-hot encoding, further described in 2.3.1 and was done for the hour-of-the-day and day-of-the-week using the attached timestamp, in addition to the id of each node and the population category. How the population of each node was categorized is further outlined below in 3.4.6.

### 3.4.6 Feature Engineering

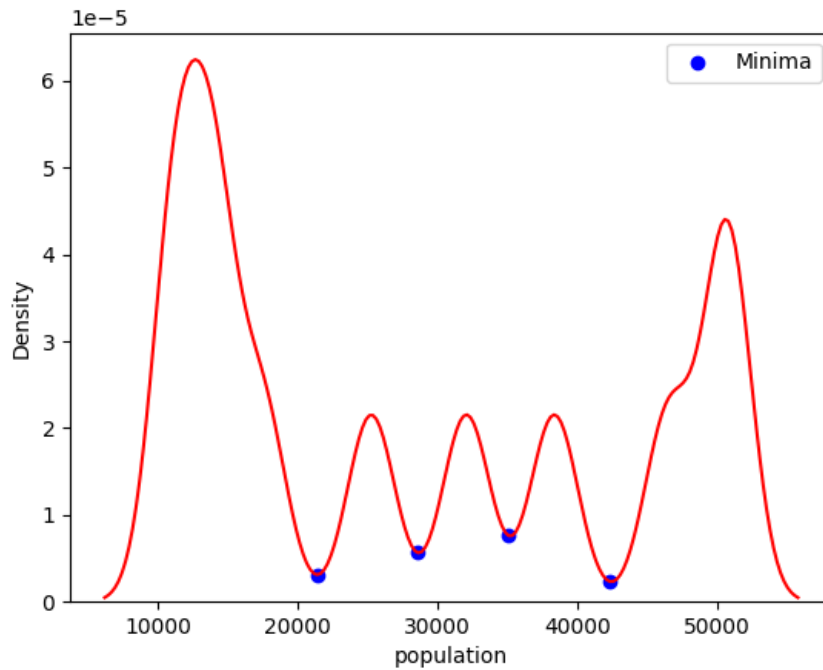
As mentioned in 3.4.1, the population was attained using an external dataset created by SSB. For each node, a mapping between the four closest  $1 \text{ km}^2$  squares and the respective node's GPS coordinates was performed. The decision to look at four squares, and not just one, is to get a more representative measure of population, especially for nodes on the edge of multiple squares. Figure 3.9 demonstrates this logic.



**Figure 3.9:** Data Engineering Population to Node.



**Figure 3.11:** Box Plot overlaid with KDE for Population based on Node.



**Figure 3.10:** KDE for Population based on Node with Minima.

The aggregated population values are further categorized using intervals. The intervals were determined through an exploratory analysis of all populations for the given nodes. To define the intervals for categorizing populations, a KDE was used, described in 2.2.3. The graph produced by the KDE was then plotted as seen in Figures 3.10 and 3.11.

The local minimas from the KDE were used as boundaries between population groups. Therefore, the dataset above, containing the eleven nodes looked at, produced a total of five distinct population groups.

Aside from the features directly extracted from the raw datasets, additional features were created by means of aggregating previous data points. This was applied to generate features based on historical RTT measurements.

In Ahmed et al., 2021, it was found that using aggregated RTT 4G data from the past five seconds yielded a reasonable accuracy. However, the team decided to create various aggregated RTT features, with the largest being ten minutes. The rationale behind this decision was to initially assess which aggregation intervals yielded strong model performance and then later potentially drop the ones which do not noticeably contribute to model performance.

Finally, Table 3.7 displays every feature properly formatted and available for ML purposes. This final compiled and engineered dataset of the eleven nodes in Oslo will from hereon be referred to as the *Centralized Dataset*.

Feature	Description
is_fault	Indicates if there's a fault
RSSI	Received Signal Strength Indication
RSRQ	Reference Signal Received Quality
RSRP	Reference Signal Received Power
RTT	RTT from previous measurement
rtt_2s_mean	RTT mean over 2 seconds
rtt_3s_mean	RTT mean over 3 seconds
rtt_4s_mean	RTT mean over 4 seconds
rtt_5s_mean	RTT mean over 5 seconds
rtt_10s_mean	RTT mean over 10 seconds
rtt_30s_mean	RTT mean over 30 seconds
rtt_1min_mean	RTT mean over 1 minute
rtt_5min_mean	RTT mean over 5 minutes
rtt_10min_mean	RTT mean over 10 minutes
ts_0, ts_1, ..., ts_23	Closest hour of the measurement timestamp
node_id_4120, node_id_4125, ...	Node IDs
population_category	Categorized population level

**Table 3.7:** List of Features in Dataset.

This section has so far discussed which features were chosen and engineered when moving forward with the dataset. However, to further determine which aspects were valuable, the team conducted a thorough exploratory data analysis, written in the next subsection 3.4.7.

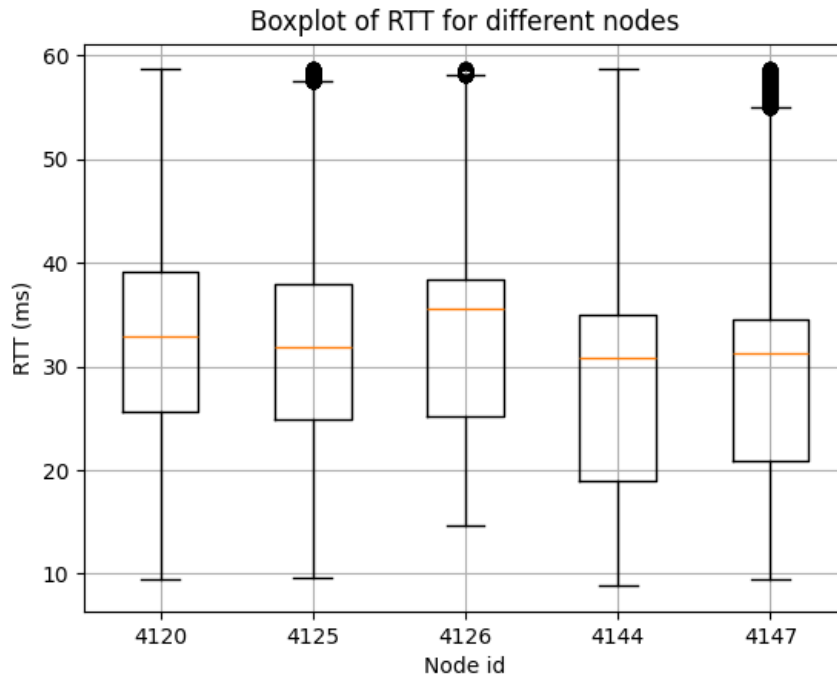
### 3.4.7 Exploratory Analysis

The data processing stages are dependent on a deeper understanding of the dataset; this understanding was attained through a comprehensive exploratory analysis. The results from the data exploration is presented in this subsection. To perform the analysis, a subset of the Centralized Dataset was used containing one week of data for five of the eleven nodes. This resulted in a manageable subset, more suiting the iterative nature of an exploratory analysis.

Initially, summary statistics of the raw measurements was generated.

	RSSI	RSRQ	RSRP	RTT
mean	-63.97	-16.69	-99.08	45.16
std	4.20	2.31	3.61	208.82
min	-78.00	-20.00	-110.00	9.52
25%	-66.00	-18.00	-101.00	27.42
50%	-63.00	-17.00	-99.00	34.23
75%	-61.00	-16.00	-96.00	42.40
max	-54.00	-9.00	-82.00	36775.70

Thereafter, a more in-depth analysis of the distributions of various features was performed. This step involved creating boxplots and cumulative distribution function graphs. These plots were further used when defining underperformance in 3.4.8.

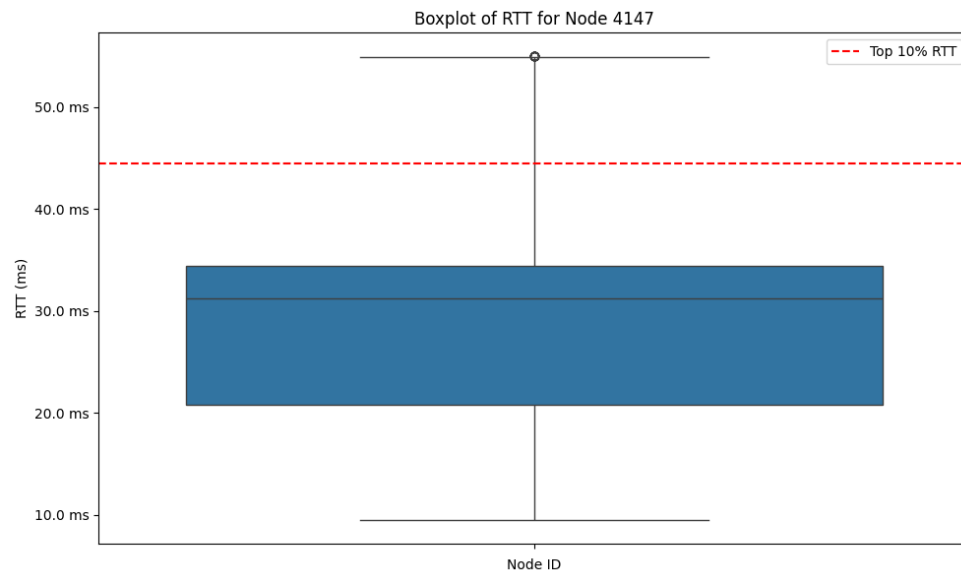


**Figure 3.12:** Boxplot for RTT of Five Nodes over One Week.

From Figure 3.12, it was clear that although nodes could vary in average RTT, the general value was somewhere between 30-40 ms.

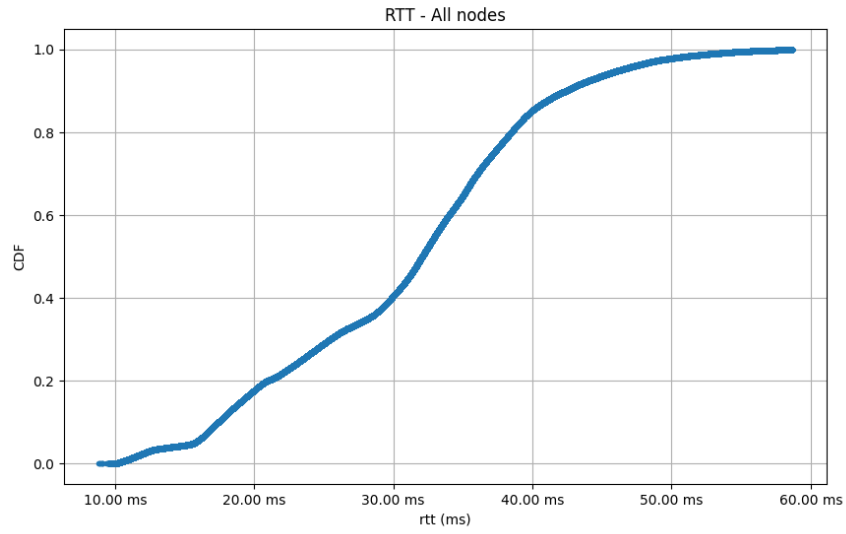
Furthermore, from the cumulative distribution function, Figure 3.14, it is clear that the data for all nodes has quite an even distribution of RTT.

Correlated features may convolute the performance and explanations of a

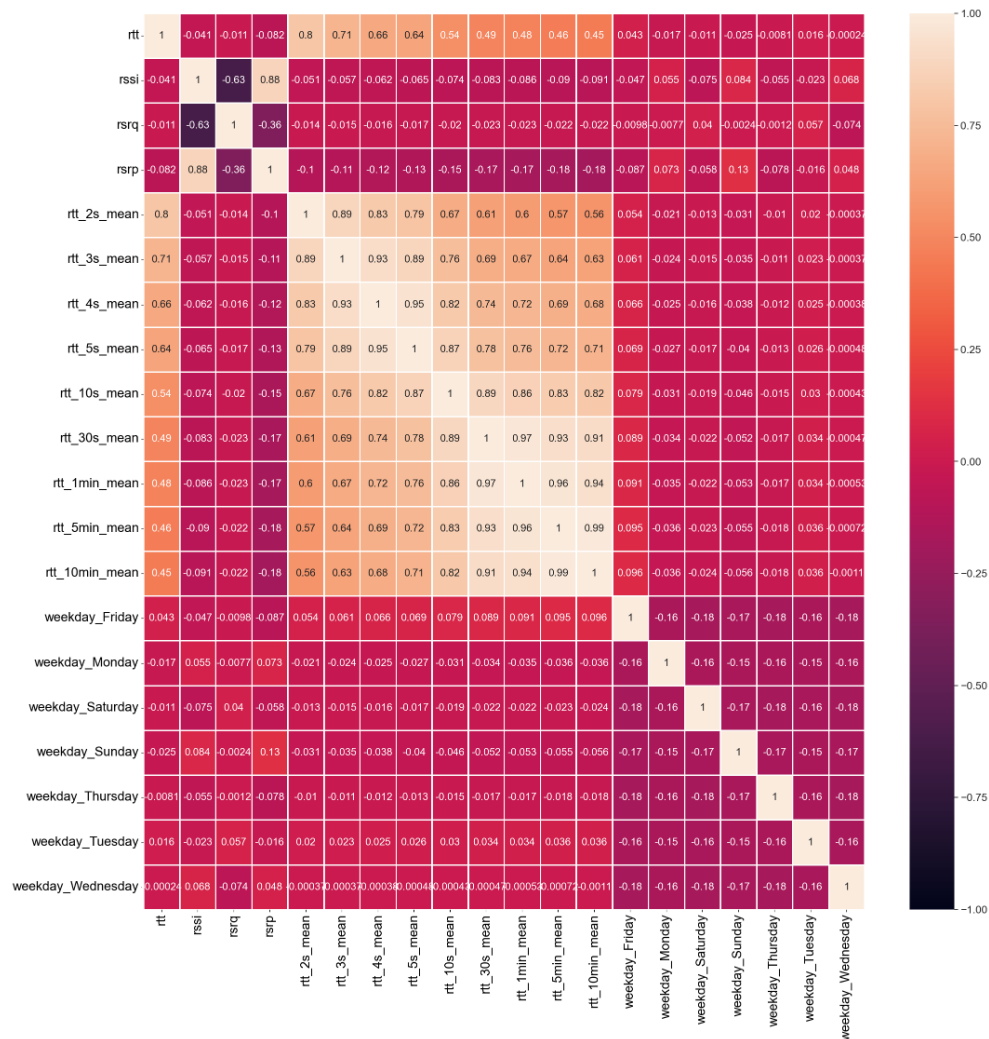


**Figure 3.13:** Boxplot of Individual Node RTT with Cut-Off for Underperformance.

model. Therefore, it was essential to conduct a correlational analysis of the features. Although non-linear correlations such as Kendall and Spearman correlation were explored, the most significant correlational matrix was for the linear correlations, i.e. Pearson's correlation. This correlation matrix can be seen in Figure 3.15.



**Figure 3.14:** Cumulative Distribution Function for RTT of Five Nodes over One Week.



**Figure 3.15:** Pearson Correlation Matrix for Features.



Due to the size of the correlation matrix, Figure 3.15 contains a subset of the original features. The features that have been removed are all the timestamp features, due to the lack of correlation.

On the other hand, multiple interesting observations can be made from the correlation matrix presented. For one, all historical RTT measurements have strong positive correlations with the preceding mean. For example, RTT has a correlation coefficient of 0.80 with mean of previous 2 seconds of RTT, *rtt\_2s\_mean*. Furthermore, all of the three passive quality measurements are correlated. RSSI and RSRP have a strong positive correlation of 0.88; this should, however, not come as a surprise after reading section 2.1.3. RSRQ and RSSI have a moderately strong negative correlation of -0.63.

From these observations, the team was made aware of collinearity of features. This relation will be important in understanding the results and explanations of the model.

### 3.4.8 Defining Underperformance in the Network

Initially, the team was told they would receive expert help with labeling faults in the dataset. This turned out not to be the case.

Due to this lack of labels linking back to actual intra-network faults, a new approach had to be considered. Recall from 2.1.3 that network underperformance can be defined as a network not operating at its expected performance. From the end-user perspective, RTT becomes a natural candidate to use as a metric for performance as it directly measures the time-lag between sending and receiving data.

Since this definition of underperformance relies on expected network performance, the initial task involves establishing the normal values for RTT. Following this, any significant deviations can be quantified and subsequently classified as underperformance.

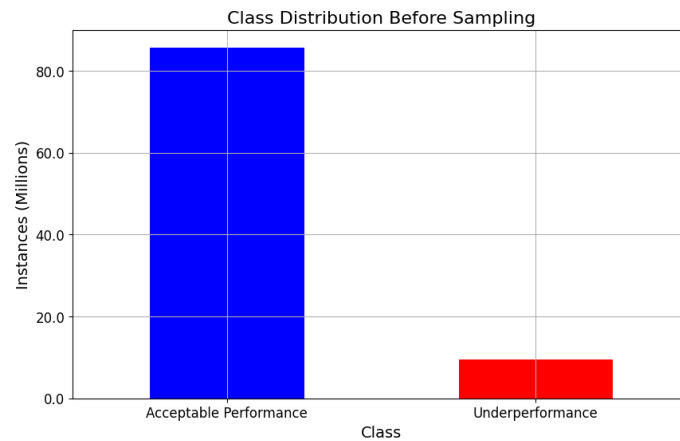
To simulate the imbalanced nature of underperformance by the user, the RTT thresholds were determined according to (Ahmed et al., 2021). In turn, this created a 9:1 split, with the minority class being underperformance. The distribution of the RTT values for the dataset is visualized in Figures 3.12, 3.13 and 3.14. This is an arbitrary threshold, but as will be mentioned in discussion 5.3, such a value is based on the concept of an SLA, as described in 2.1.2.

Finally, these underperformance / not underperformance labels are generated based on the threshold for each one-second RTT value. However, since this will be a predictive model, the model should always predict the

label for the measured RTT one second ahead. As a result, all labels are shifted down by one entry in the dataset. A consequence of this is that the model will predict if underperformance occurs exactly one second in advance. It will not, however, predict if underperformance occurs between the time of the prediction and the next second, as this would require a completely different dataset.

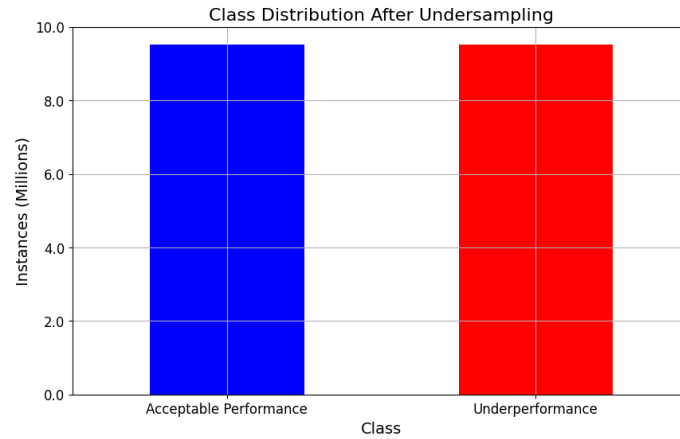
### 3.4.9 Addressing Data Imbalance

Defining underperformance to be the highest 90th percentile resulted in a heavily imbalanced dataset (9:1), displayed in Figure 3.16. This imbalance poses concerns when training a ML model as outlined in 2.3.1.



**Figure 3.16:** Class Distribution Before Sampling.

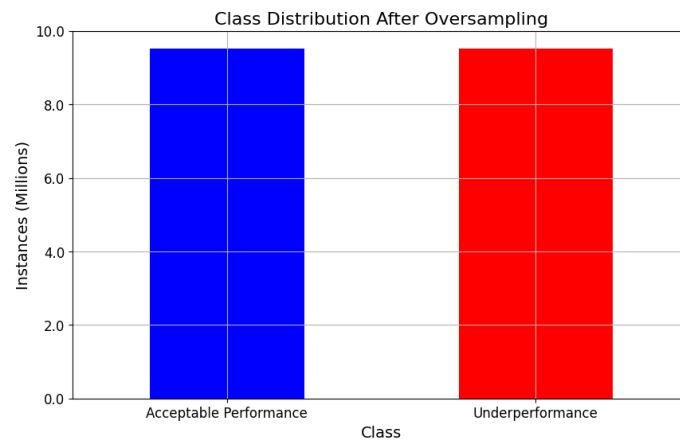
To balance the dataset for better model performance, two main strategies were employed. Firstly, undersampling was performed, as described in 2.3.1. Both the NearMiss and random undersampling methods from the imblearn Python package were tested. Figure 3.17 displays the class distribution in the dataset after undersampling. During the undersampling, the goal was to achieve class parity. This means both classes contain the same amount of instances.



**Figure 3.17:** Class Distribution After Undersampling.

Ahmed et al., 2021 displays that SMOTE showed positive results when using data from NNE. Therefore, SMOTE along with other oversampling methods were tested. The oversampling methods tested were the ones displayed in Table 2.1. To be able to test every method within the time requirements, a subset of the Centralized Dataset was used containing one node with two weeks of measurements.

As with the undersampling methods, class parity was again the chosen goal. Figure 3.18 displays the distribution after oversampling, which differs from Figure 3.17 by having a larger minority class.



**Figure 3.18:** Class Distribution After Oversampling.

Random undersampling ended up being used moving forward, as will be further discussed in 5.1.5.

### 3.4.10 Model Choice

In order to determine which model to use when carrying out experiments, the team reviewed related work and constructed a list of potential models. To optimize each model to best fit the Centralized Dataset, grid-search was performed using the search parameters from Table 4.1. Since the dataset is imbalanced, the grid-search was stratified due to the reasons outlined in 2.3.4. Using the results from the grid-search, a model was chosen to perform the subsequent experiments with.

Model name	Hyperparameter space
Ada Boost	Number of Estimators: [10, 20, 50] Algorithm: [SAMME, SAMME.R] Learning Rate: [0.1, 0.01, 0.05]
Cat Boost	Depth: [7, 8, 9] Learning Rate: [0.3, 0.1, 0.01] Iterations: [1500, 2000, 2500] Subsample: [0.6, 0.8, 1.0] Bootstrap Type: [Bernoulli, MVS] L2 Leaf Reg.: [1, 3, 5]
Decision Tree	Criterion: [Gini, Entropy] Max Depth: [None, 5, 10, 15] Min Samples Split: [2, 5, 10] Min Samples Leaf: [1, 2, 4]
LightGBM	Max Depth: [3, 4, 5] Num Leaves: [20, 30, 40] Learning Rate: [0.01, 0.1, 0.3] Number of Estimators: [50, 100, 150] Subsample: [0.6, 0.8, 1.0] Colsample Bytree: [0.6, 0.8, 1.0]
Logistic Regression	C : [0.001, 0.01, 0.1, 1, 10, 100] Penalty: [L1, L2]
RF	Number of Estimators: [50, 100] Max Depth: [None, 10] Min Samples Split: [2, 5] Min Samples Leaf: [1, 2]
SVC	Kernel: [linear, poly, rbf, sigmoid] C: [0.1, 1, 10] Degree: [1, 2, 3] Gamma: [0.01, 0.1, 1]
XGBoost	Max Depth: [3, 4, 6] Learning Rate: [0.01, 0.1, 0.2] Number of Estimators: [50, 100, 150, 200] Subsample: [0.6, 0.8] Colsample Bytree: [0.6, 0.9]

**Table 3.8:** Hyperparameter Space for Each Model.

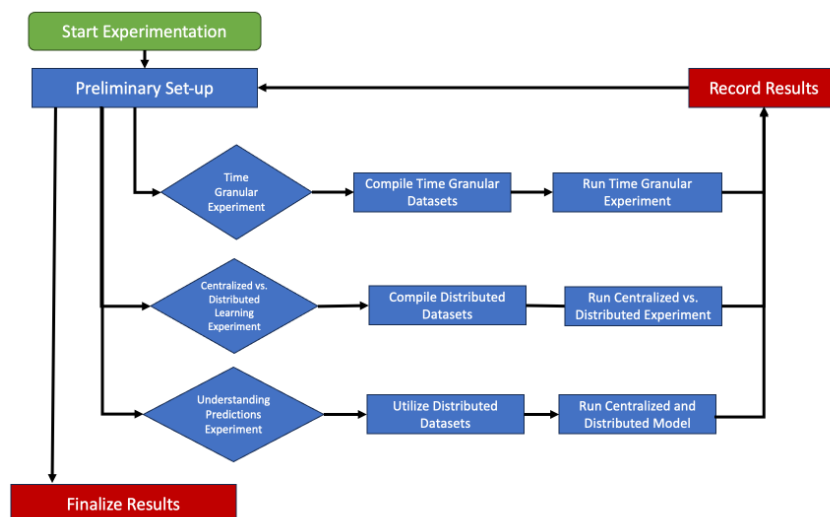
As can be seen in Table 4.1, SVC was originally one of the candidate models. Regrettably, SVC had to be removed as a candidate because the grid-search did not finish within the project time frame.

### 3.4.11 Validation Metrics

When comparing model performance, for example during a grid-search, the F1-score was utilized. However, when comparing performance more holistically, additional metrics such as AUC, TP-rate, FP-rate, and AUPRC were also factored in, although to a lesser degree. While not validation metrics, factors like disk space usage, compute time per prediction, and train duration were also assessed. The rationale behind this is left to the discussion section 5.1.1.

## 3.5 Conducting Research Experiments

In this section, the process for conducting the specific research experiments will be reviewed. The methodology described in Section 3.4 provides the starting point for all the experiments. Figure 3.19 detail how each experiment were conducted and how they all are based of the preliminary set-up.



**Figure 3.19:** Flow Chart of How the Experiments Were Carried Out.

### 3.5.1 Data Granularity Experiment

As questioned in 1.2, finding out how modulating data measurement granularity affected model performance was one of the research questions for this thesis. This experiment was not feasible to perform on the entire Centralized Dataset. Consequently, this experiment is performed on one select node from the Centralized Dataset.

#### Selection of the Node

The NNE node from the Centralized Dataset with the most uptime was chosen for further investigation. This was node 4144.

#### Processing

In total, nine datasets were processed to create the time granularities of 1, 2, 3, 4, 5, 10, 15, 30, and 60 seconds. For this experiment, the engineered RTT mean features had to be recalculated based on the respective granularities for the dataset. Further, the dataset underwent feature engineering as described in 3.4.6. Although, instead of each granularity dataset having their own RTT-threshold, the threshold from the original dataset was used across all granularities.

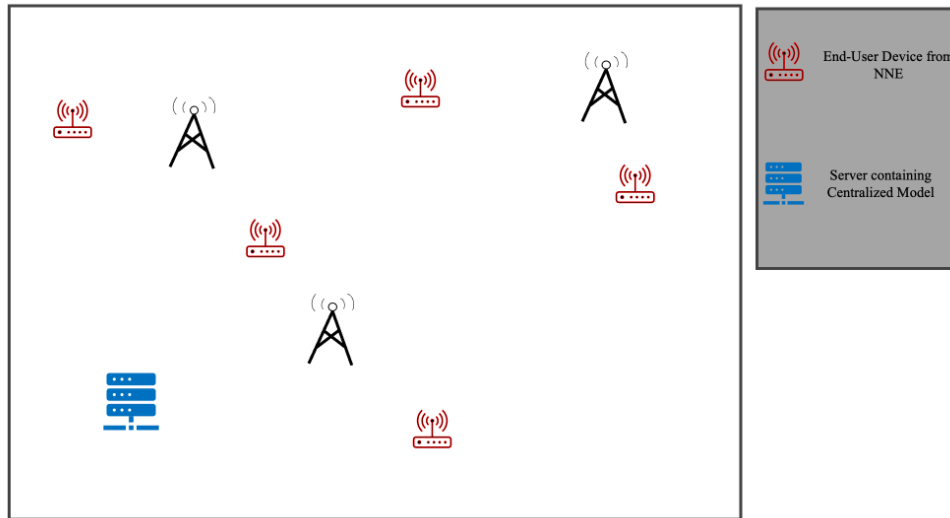
#### Generating Results

To generate the results for the granular datasets, the datasets were all run on the chosen model from 3.4.10 with optimal parameters from the stratified grid-search. The results from running the models was then saved, and graphs visualizing the performance metrics were created, which can be seen in 4.2.

### 3.5.2 Centralized vs. Distributed Learning Experiment

Originally, it was hypothesised that variance in each NNE node warranted a separate model for each node. However, after conducting the exploratory analysis on the data (see 3.4.7), the uniform distribution and similarity between nodes' values suggested the possibility of using a centralized model. Therefore, another experiment the team conducted was comparing the performance of a centralized model and distributed models for the Centralized Dataset.

The centralized model benefited from additional information such as node id and exogenous features such as the population category for the respective node.



**Figure 3.20:** Diagram Representing The Network Landscape with Differing Models.

### Processing

Since the Centralized Dataset contained all eleven nodes in Oslo, for the distributed models, additional datasets based on this had to be created. The distributed datasets were generated through filtering the original raw data and compiling the same three months worth of data, but separated for each of the eleven nodes.

The distributed datasets were handled similarly as in 3.4.5. One difference between the centralized dataset and the distributed dataset was the features. Some of the categorical features in the centralized dataset were no longer relevant on an individual basis; these include node id and population.

### Generating Results

For each node and for the centralized dataset, the same chosen model from 3.4.10 was run. The same validation metrics as mentioned in 2.3.2.2 were utilized.

To analyze the difference between the model types, two different validation experiments were run. For the first experiment, the total model performance on the respective datasets was validated and compared. However, it was also interesting to see how the centralized model performed on a node-level. Therefore, the centralized model was validated on subsets of the test set, filtered using the node id.

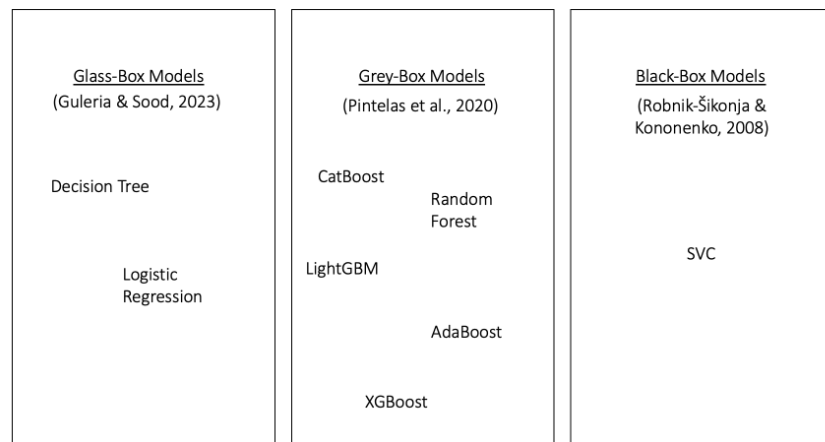
### 3.5.3 Understanding Model Output

When analysing the outputs from the model, the temporal dimension of model predictions becomes interesting. To do this, both the centralized model and one of the distributed models from the previous section was further investigated. For the distributed model, the same node as chosen in the granularity experiments was utilized, and for the same reasons.

An important part of understanding the outputs is understanding where the model went wrong. Therefore, an analysis of the mispredictions was conducted. In this case, FNs are of particular significance since correctly classifying positives is one of the main concerns of the model.

### 3.5.4 Tackling explainability of the model

The chosen model for the experiments was the RF model. This choice of model will later be discussed in 3.4.10. According to Pintelas et al., 2020, this model falls under the grey-box category of explainability. As such, post-hoc techniques are necessary to understand why it produces a certain output.



**Figure 3.21:** Explainability Categorization of Chosen Models.

The post-hoc techniques chosen are two model-agnostic techniques, PaP and SHAP values, and one-model specific to tree-based models, MDI. These methods were run on the centralized and one distributed RF model. Since all of these models produce a list of feature importance, a direct comparison can be made between them. Still, it is important to keep in



mind the logical differences and effects that feature relationships have on the produced results.

For efficiency, the tree-explainer (S. M. Lundberg et al., 2019) implementation for SHAP values was utilized to calculate local and global feature importance. Due to the size of the models, dataset, and tree-explainer, samples of the test set were utilized to explore feature importance. Furthermore, for global feature importance, it was essential to maintain a representative distribution and therefore sklearn's `StratifiedSampleSplit` was used for sampling.



## 4. Results

This chapter presents the findings from the ML experiments that have been described in the method section. The focus of this research has been to evaluate various predictive models in identifying underperformance in 5G cellular networks. In this chapter, the results presented are therefore important for understanding both the strengths and limitations of the tested ML models in real-world scenarios.

### 4.1 Model Comparison

This section displays the results comparing the various ML models listed below. Recall for the comparison, the Centralized Dataset containing eleven NNE nodes in the Oslo Metropolitan Area was utilized. First, the grid-search results for all models will be presented, which will make up the best estimators for each model. Then, the results using these best estimators will be presented. Finally, the results from the sampling experiments will be displayed.

The seven models compared are:

1. AdaBoost
2. CatBoost
3. Decision Tree
4. LightGBM
5. Logistic Regression
6. RF
7. XGBoost

#### 4.1.1 Grid-Search

The results in Table 4.1 present the best hyperparameters along with their corresponding search values. Remember, the search space for the grid-search was defined in 3.8, and the metric used to determine model performance was the F1-Score.

Model name	Best hyperparameter
Ada Boost	Number of Estimators: 20 Algorithm: SAMME.R Learning Rate: 0.1
Cat Boost	Depth: 9 Learning Rate: 0.3 Iterations: 2500 Subsample: 0.8 Bootstrap Type: MVS L2 Leaf Reg.: 3
Decision Tree	Criterion: Entropy Max Depth: 15 Min Samples Split: 5 Min Samples Leaf: 2
LightGBM	Max Depth: 5 Num Leaves: 40 Learning Rate: 0.3 Number of Estimators: 150 Subsample: 0.6 Colsample Bytree: 1.0
Logistic Regression	C : 0.1 Penalty: L2
RF	Number of Estimators: 100 Max Depth: None Min Samples Split: 5 Min Samples Leaf: 2
XGBoost	Max Depth: 6 Learning Rate: 0.2 Number of Estimators: 200 Subsample: 0.8 Colsample Bytree: 0.9

**Table 4.1:** The Best Values for Each Hyperparameter After Grid-Search.

#### 4.1.2 Comparison Using Best Estimator

Table 4.2 outlines every validation metrics for all the different models. The best value for each metric is highlighted in bold.

Model Name	Precision	F1 Score	AUC	AUPRC	TNR/Specificity	FPR	FNR	TPR/Recall
AdaBoost	0.419	0.559	0.855	0.367	0.870	0.130	0.161	0.839
CatBoost	<b>0.465</b>	<b>0.610</b>	<b>0.886</b>	<b>0.432</b>	<b>0.887</b>	<b>0.113</b>	0.115	0.885
Decision Tree	0.326	0.465	0.812	0.283	0.814	0.186	0.189	0.811
LightGBM	0.417	0.560	0.860	0.370	0.868	0.132	0.148	0.852
Logistic Regression	0.130	0.230	0.627	0.130	0.268	0.732	<b>0.013</b>	<b>0.987</b>
RF	0.448	0.592	0.876	0.403	0.881	0.119	0.130	0.870
XGBoost	0.417	0.560	0.860	0.370	0.867	0.133	0.147	0.853

**Table 4.2:** Performance Metrics for the Best Estimator for Each Model.

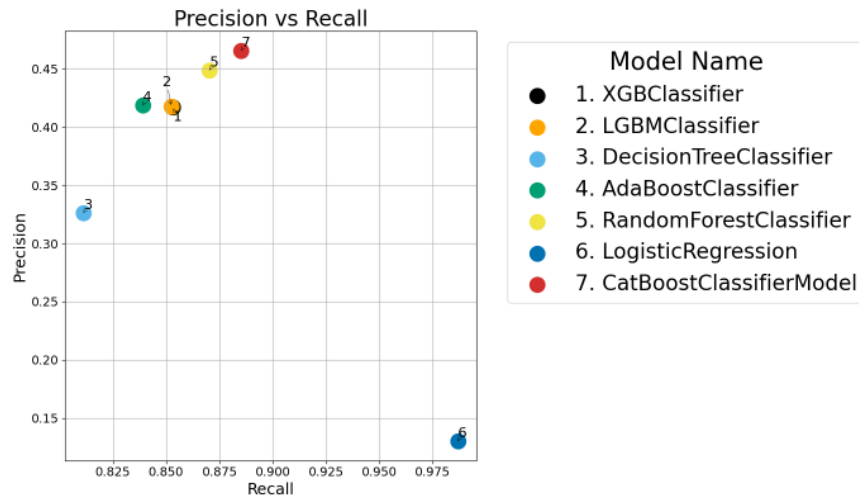
From these results, CatBoost outperforms the other models on all metrics

except recall/TP-Rate and FN-Rate. For TP-Rate and FN-Rate, logistic regression performs the best. In other words, the logistic regression model is very effective at classifying instances of underperformance as underperformance. CatBoost is the most well-rounded model when it comes to performance. RF comes in second with only slightly worse performance across every metric.

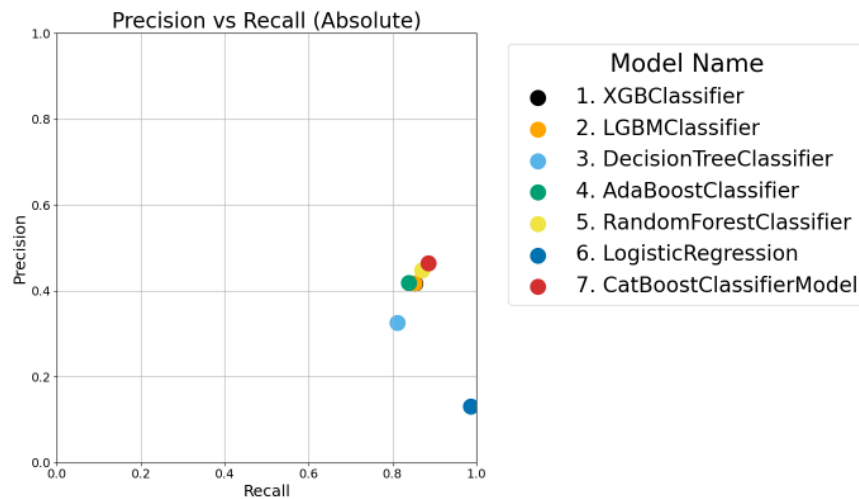
To further understand these results, a series of plots will be presented to compare the ML models using their best estimator. Each subsequent figure will share the same format, being a close-up plot of all the ML models accompanied by another plot displaying the metrics from an absolute point of view. This makes sure any bias from small value ranges are removed, and at the same time, it will be easy to see if the performance is similar. This same format of plots will be reused in 4.2 and partly in 4.3.

### **Precision vs Recall**

The first observation to make is that logistic regression performs notably worse than the other models at precision, while scoring significantly better at recall. When looking at 4.1b, it becomes evident that all models other than logistic regression are grouped closely together and the difference in precision and recall in absolute terms is not profoundly large. Reading from the diagram, CatBoost and RF perform the best, with CatBoost outperforming RF by about 0.02 in both precision and recall.



(a) Close-up Plot of Precision vs Recall.



(b) Absolute Plot of Precision vs Recall.

**Figure 4.1:** Scatter Plots Displaying Precision vs Recall for Each Model-Compare Measurement. Top Right is Better.

### Computational Cost

The computational cost for each of the models was measured and is displayed in the Table 4.3. Logistic regression has by far the the smallest disk size and is also faster than the all other models at performing predictions. At the same time, logistic regression is one of the slowest models to train, with AdaBoost being the only model that is slower. XGBoost and LightGBM are the models that have the lowest train times, having 56 and 71 seconds respectively. Additionally, it is worth mentioning they both use low disk space, having model sizes lower than 200KB. RF has an extremely

large disk space usage at around 17GB, making it almost 46 times larger than the next largest model, decision tree, which itself is almost 388MB. CatBoost has both a relatively small disk size and also a low time for compute per analysis, but at the cost of taking over 700 seconds to train.

Model Name	Compute per Analysis (s)	Disk (MB)	Train Time (s)
AdaBoost	5.438e-06	0.029	3234
CatBoost	9.314e-07	20.779	707
Decision Tree	1.220e-06	387.910	201
LightGBM	1.441e-06	0.196	71
Logistic Regression	5.928e-07	0.002	2993
RF	7.335e-06	17733.540	390
XGBoost	2.626e-06	0.176	56

**Table 4.3:** Performance Metrics for the Best Estimator for Each Model.

### Sampling Experiment

The results from the testing of sampling methods, as written in 3.4.9, are described here. The Table 4.4 denotes not only the performance metrics, but also time requirements needed for using the different methods.

In the table, it is seen that the oversampling methods cause an almost doubling of training time compared to no sampling and an almost tenfold increase in the training time compared to random undersampling.

Random oversampling achieved the highest F1-score of 0.641, showing a strong balance between precision and recall, being 0.5645 and 0.7414, respectively. Borderline SMOTE and AdaSyn had slightly lower F1-scores of 0.6251 and 0.6277 respectively, in addition to requiring more training time. Random undersampling stood out with the highest AUC of 0.8611 and the highest recall of 0.8826, combined with the shortest training and sampling times (82 seconds and 0.20 seconds). SMOTE has no performance advantages over other methods and is worse than random oversampling in all metrics along with having the longest training time of 739 seconds. The model without any sampling (None) had a F1-score of 0.5783 and AUC of 0.7445, the second lowest of all methods in both regards. NearMiss, despite having a high recall, had the worst F1-score of 0.3224 and the worst AUC of 0.6729.

Model Name	Precision	F1 Score	AUC	TNR/Specificity	TPR/Recall	Train Time (s)	Sampling Time (s)
AdaSyn	0.5486	0.6277	0.8219	0.9103	0.7335	693.9073	25.7258
Borderline SMOTE	0.5434	0.6251	0.8219	0.9081	0.7356	719.3431	24.2212
NearMiss	0.1978	0.3224	0.6729	0.4755	0.8704	93.0940	19.0330
None	<b>0.6274</b>	0.5783	0.7445	<b>0.9527</b>	0.5363	427.1795	<b>0</b>
Random Undersampling	0.4500	0.5960	<b>0.8611</b>	0.8397	<b>0.8826</b>	<b>82.3643</b>	0.1984
Random Oversampling	0.5645	<b>0.6410</b>	0.8282	0.9150	0.7414	590.7924	3.6753
SMOTE	0.5508	0.6321	0.8258	0.9101	0.7414	739.1876	4.6086

**Table 4.4:** Performance Metrics of RF with Various Resampling Methods.

## 4.2 Time Granular Measurements

This sections will give an overview of the results from datasets with varying time granularities. The specific dataset and variations in time granularities are described in 3.5.1. Effects of measurement frequency on the chosen model will be explored, further displayed in five graphs. The results are displayed in Table 4.5.

Granularity (s)	Precision	F1 Score	AUC	AUPRC	TNR/Specificity	FPR	FNR	TPR/Recall
1	<b>0.393</b>	<b>0.541</b>	0.860	<b>0.355</b>	0.850	0.150	0.129	0.871
2	0.368	0.520	0.862	0.336	0.841	0.159	0.117	0.883
3	0.365	0.517	0.862	0.334	0.840	0.160	0.117	0.883
4	0.363	0.515	0.861	0.332	0.839	0.161	<b>0.116</b>	<b>0.884</b>
5	0.382	0.533	<b>0.865</b>	0.348	<b>0.851</b>	<b>0.149</b>	0.120	0.880
10	0.371	0.522	0.863	0.337	0.848	0.152	0.122	0.878
15	0.366	0.515	0.857	0.330	0.846	0.154	0.133	0.867
30	0.360	0.509	0.854	0.326	0.837	0.163	0.129	0.871
60	0.359	0.504	0.843	0.318	0.842	0.158	0.157	0.843

**Table 4.5:** Performance Metrics for the Random Forest Model Across Different Time Granularities.

The 1-second granularity performs well across all metrics, achieving the highest F1-score of 0.541 and the strongest AUPRC of 0.355. This means the 1-second granularity is the best granularity at balancing correct predictions of underperformance with few false predictions. Following, the 2-, 3-, and 4-second granularities exhibit similar performance to the 1-second granularity, with lower AUPRC and higher recall, but comparable AUC values around 0.861-0.862.

The 5-second granularity shows the highest AUC of 0.865 and specificity of 0.851, while maintaining relative performance in other metrics with an F1-score of 0.533 and an AUPRC of 0.348. The 10-second granularity continues to perform well with an AUC of 0.863 and recall of 0.878.

Performance begins to decline with the 15-second granularity, which shows a lower AUC of 0.857 and an F1-score of 0.515. Further, the 30-second granularity further decreases with an AUC of 0.854 and an F1-score of 0.509. The 60-second granularity exhibits the lowest performance across the board, with an F1-score of 0.504, an AUC of 0.843, and an AUPRC of 0.318.

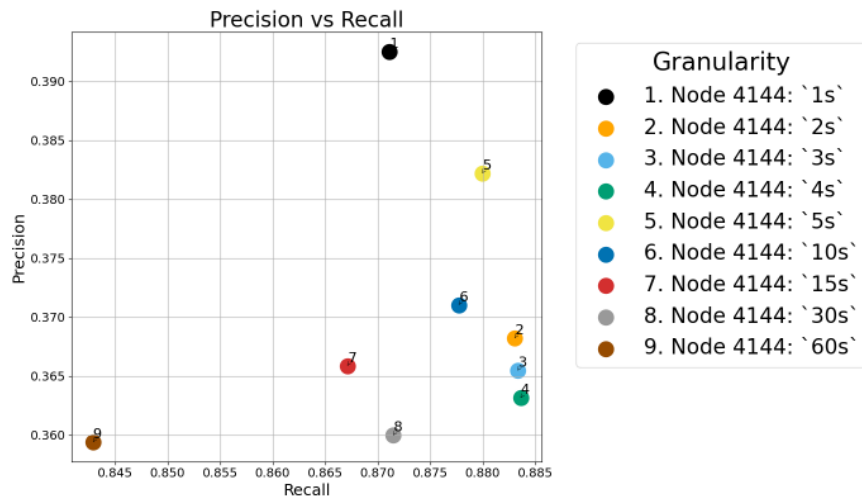
Although there is a performance decrease as the granularities increase, the difference in AUPRC and F1-score from the best to the worst is only 0.037 for both. In addition, the other metrics do not have a drastic drop off. The metric with the largest difference is recall, being 0.041 from the 4-second granularity to the 60-second granularity.

To better display the values in the table, two scatter plots comparing met-

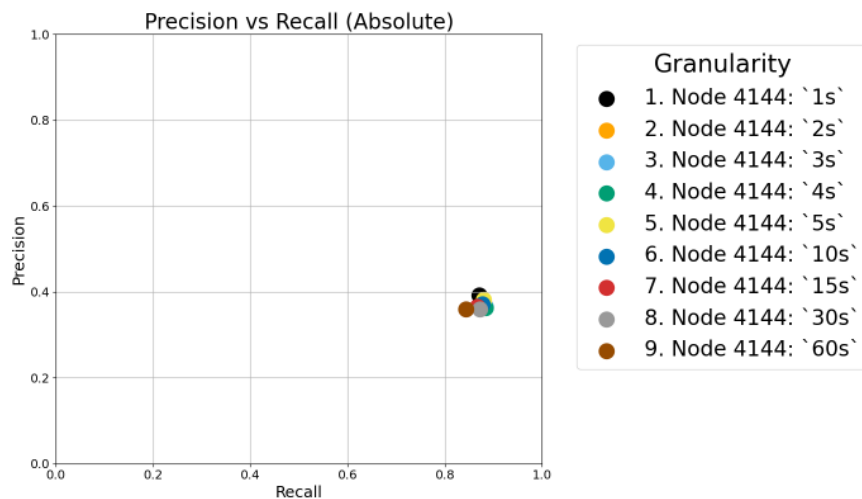


rics are described below.

Figure 4.2 includes plots that show the performance of each granularity for the chosen node when comparing precision and recall. The first plot, Figure 4.2a, shows the lower granularities of 15-, 30-, and 60-seconds all have worse recall and precision than the rest. Although, when seen from an absolute point of view in Figure 4.2b, it is clear that all granularities are relatively similar in performance.



**(a)** Close-Up Plot of Precision vs Recall.

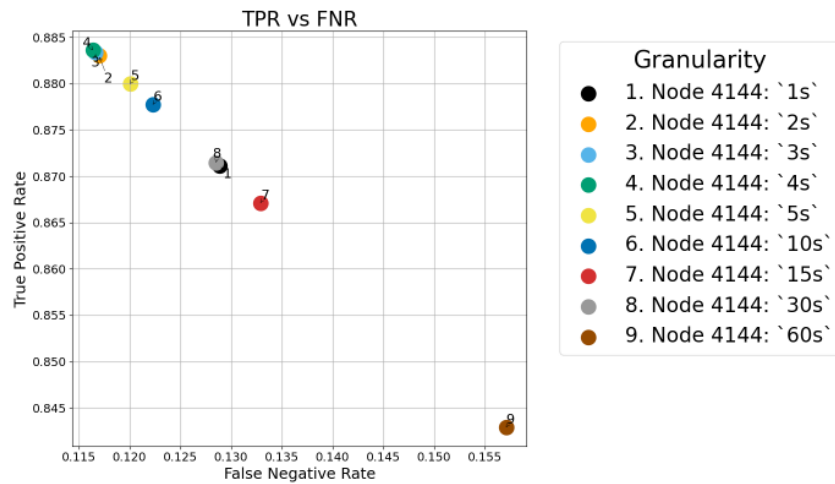


**(b)** Absolute Plot of Precision vs Recall.

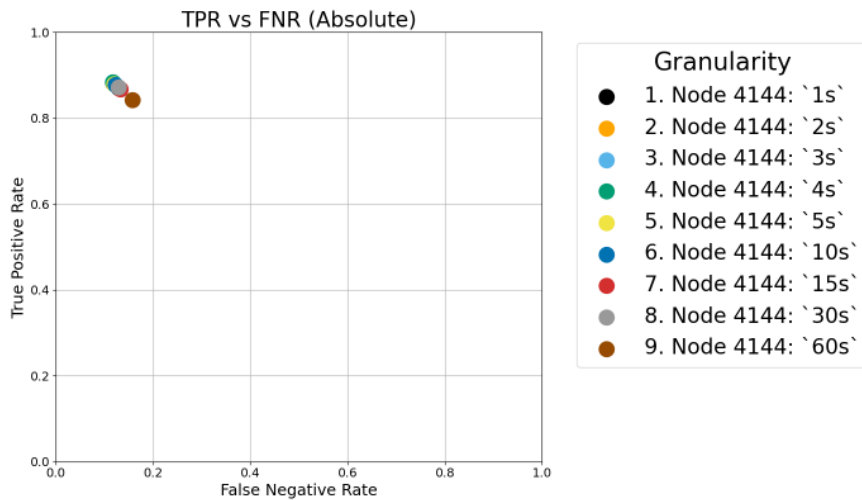
**Figure 4.2:** Scatter Plots displaying Precision vs Recall for each Granularity Measurement. Top Right is Better.

Figure 4.3 displays plots comparing the TP-Rate and the FN-Rate. This plot

shows all granularities in a line because these two rates combined will always become 1. Three distinct groups can be seen on the plot in Figure 4.3a. The first group displays the granularities performing best which includes the 2-, 3-, 4-, 5- and 10-second granularities. Further, the 1-, 30-, and 15-second granularities are displayed grouped together and have similar performance in this aspect. Finally, the 60-second granularity scores much lower. This is still visible in the absolute plot 4.3b, where the 60-second granularity has separation from the other marks on the plot.



(a) Close-Up Plot of TP-Rate vs FN-Rate.

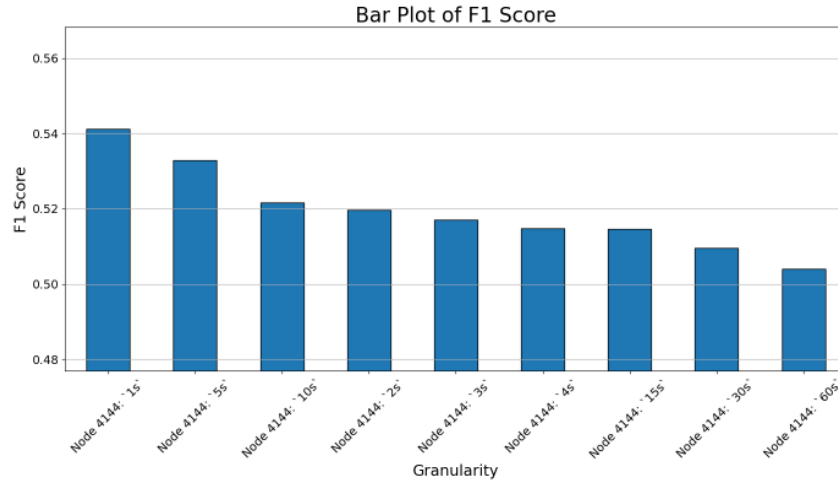


(b) Absolute Plot of TP-Rate vs FN-Rate.

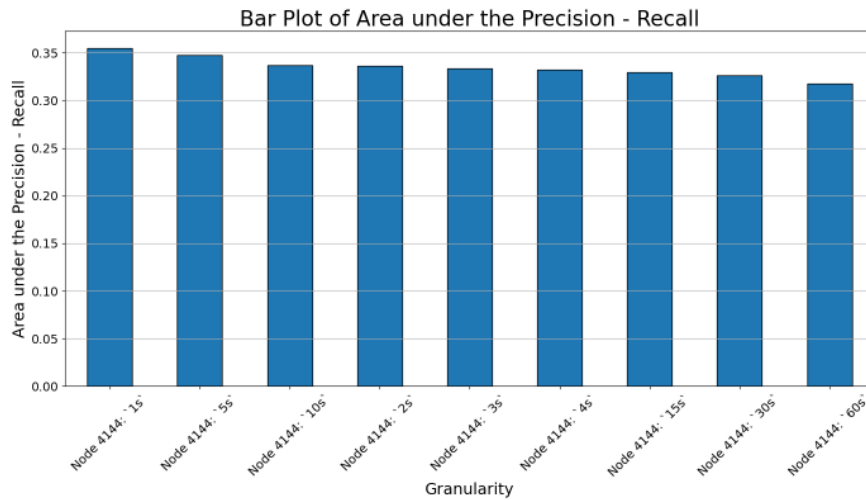
**Figure 4.3:** Scatter Plots Displaying TP-Rate vs FN-Rate for each Granularity Measurement. Top Left is Better.

Further, two bar charts will display a comparison of the varying granular-

ities for the F1-score and AUPRC. These charts are both relative.



**Figure 4.4:** Bar Chart Comparing F1 Score for Each Granularity Measurement. Higher is Better.



**Figure 4.5:** Bar Chart Comparing AUPRC Score for each Granularity Measurement. Higher is Better.

Even with a relative y-axis, a small difference in the values for both the F1-score and AUPRC can be observed for the different granularities. That said, 1-second granularity performs the best in both metrics with 60-second performing the worst.

### 4.3 Centralized vs. Distributed Models

In the following section, the results from the experiments comparing a centralized model vs. distributed models will be presented. In the first three plots, the general performance of the centralized model is compared with the individual node models. Thereafter, a more fine-grained view of performance is undertaken through five plots showing performance per node.

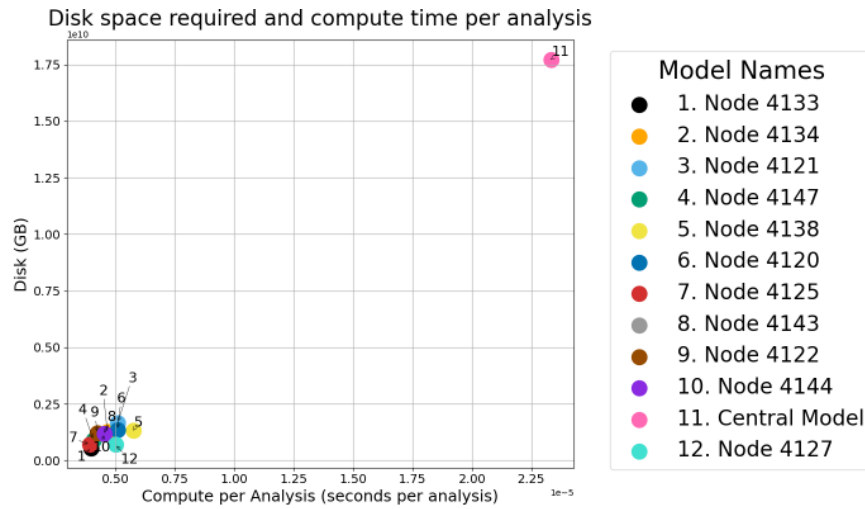
#### 4.3.1 General Performance

In the general performance experiment, the results from validating the centralized and distributed models using their respective test sets in entirety are presented. All of the data from this experiment is shown in Table 4.6.

Model Name	Precision	F1 Score	AUC	AUPRC	TNR/Specificity	FPR	FNR	TPR/Recall	Disk Usage (MB)
Node 4120	0.2332	0.3550	0.7354	0.1990	0.7281	0.2719	0.2572	0.7428	1297
Node 4121	0.2899	0.3886	0.7146	0.2119	0.8399	0.1601	0.4107	0.5893	1583
Node 4122	0.3045	0.4537	0.8320	0.2820	0.7741	0.2259	0.1100	0.8900	1160
Node 4125	0.3360	0.4917	0.8573	0.3164	0.7978	0.2022	0.0831	0.9169	676
Node 4127	0.3146	0.4647	0.8373	0.2906	0.7861	0.2139	0.1115	0.8885	666
Node 4133	0.4240	0.5892	<b>0.9100</b>	<b>0.4127</b>	0.8548	0.1452	<b>0.0347</b>	<b>0.9653</b>	<b>514</b>
Node 4134	0.2335	0.3605	0.7514	0.2054	0.7130	0.2870	0.2102	0.7898	1201
Node 4138	0.2366	0.3602	0.7425	0.2029	0.7313	0.2687	0.2462	0.7538	1256
Node 4143	0.3452	0.4955	0.8461	0.3152	0.8149	0.1851	0.1226	0.8774	1096
Node 4144	0.3916	0.5404	0.8606	0.3541	0.8496	0.1504	0.1284	0.8716	1130
Node 4147	0.4125	0.5729	0.8947	0.3929	0.8520	0.1480	0.0625	0.9375	869
Central	<b>0.4486</b>	<b>0.5920</b>	0.8755	0.4033	<b>0.8811</b>	<b>0.1189</b>	0.1302	0.8698	16892

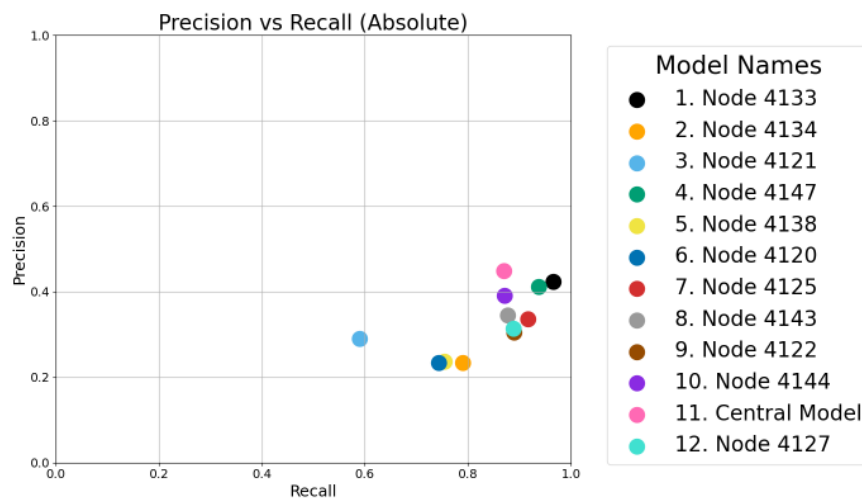
**Table 4.6:** Centralized vs. Distributed Model Results.

Figure 4.6 shows the clear increase in both disk usage and compute per analysis that a centralized model has compared to distributed models. As seen in Table 4.6, the centralized model has a disk usage of 16.892 GB, while the next highest distributed model (Node 4121) only uses 1.583 GB. Similarly, the central model has a compute time per analysis of 2.3319e-05 seconds per analysis, while the next slowest model (Node 4139) uses 5.7535e-06 seconds per analysis.



**Figure 4.6:** Scatterplot Showing Disk Space Against Compute Time per Analysis for the Centralized and Distributed Models.

Furthermore, in Figure 4.7, the differences in precision and recall between the models is highlighted. Here, the centralized model can be seen as having the highest precision with 0.4486. According to Table 4.6, with a recall of 0.8698, the central model has a higher recall than node 4134, 4138, 4120, and 4121. Still, node 4133 has the highest recall with 0.9653 and therefore lowest FN-rate with 0.0347, followed by node 4147 with a recall of 0.9375. The lowest recall belongs to node 4121 with 0.5893.



**Figure 4.7:** Scatterplot Showing Recall vs. Precision for the Centralized and Distributed Models.

### 4.3.2 Node-specific Performance

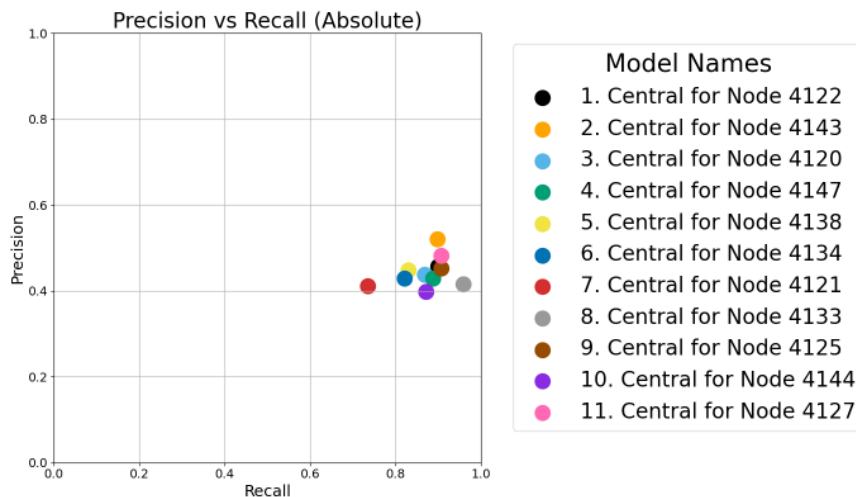
Node-specific performance looks at the centralized model's performance on a subset of the test data; more specifically, the subsets are filtered to include data from only one node. In this way, the plots below show the performance of the centralized model for each node. Later, this performance is then compared against the distributed models.

In the table below, the data from the validation of the centralized model per node is presented.

Model Name	Precision	F1 Score	AUC	AUPRC	TNR/Specificity	FPR	FNR	TPR/Recall
Central for Node 4120	0.4385	0.5826	0.8368	0.4001	0.8060	0.1940	0.1324	0.8676
Central for Node 4121	0.4118	0.5276	0.8384	0.3160	<b>0.9429</b>	<b>0.0571</b>	0.2662	0.7338
Central for Node 4122	0.4575	0.6062	0.8817	0.4224	0.8653	0.1347	0.1018	0.8982
Central for Node 4125	0.4528	0.6038	0.8936	0.4194	0.8813	0.1187	0.0941	0.9059
Central for Node 4127	0.4826	0.6300	0.8850	0.4492	0.8630	0.1370	0.0930	0.9070
Central for Node 4133	0.4166	0.5805	<b>0.9188</b>	0.4023	0.8801	0.1199	<b>0.0425</b>	<b>0.9575</b>
Central for Node 4134	0.4295	0.5639	0.8256	0.3767	0.8303	0.1697	0.1791	0.8209
Central for Node 4138	0.4484	0.5820	0.8295	0.3962	0.8299	0.1701	0.1708	0.8292
Central for Node 4143	<b>0.5202</b>	<b>0.6586</b>	0.8898	<b>0.4795</b>	0.8825	0.1175	0.1028	0.8972
Central for Node 4144	0.3987	0.5471	0.8979	0.3544	0.9245	0.0755	0.1286	0.8714
Central for Node 4147	0.4299	0.5790	0.9069	0.3877	0.9273	0.0727	0.1135	0.8865

**Table 4.7:** Centralized Model Performance per Node.

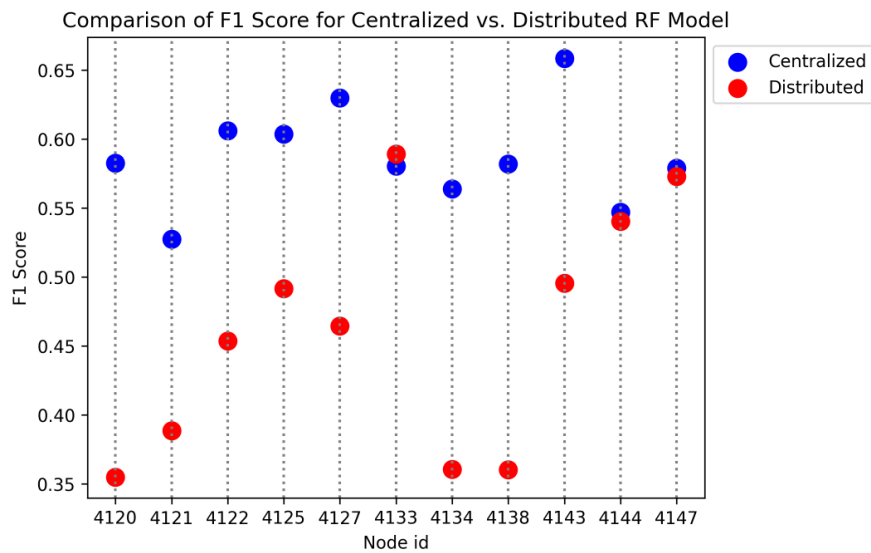
Figure 4.8 displays the precision and recall for each node subset under the centralized model. Observe how despite utilizing the same model, there is a spread in recall of 0.2237 and a spread of 0.1215 for precision. However, when looking at Figure 4.7, the distributed node models show a spread of 0.3760 for recall and 0.1908 for precision. This shows the central model has a smaller spread for these performance metrics when compared to the distributed models.



**Figure 4.8:** Scatterplot Showing Precision Against Recall for the Centralized Model, Validated against Nodes Individually.

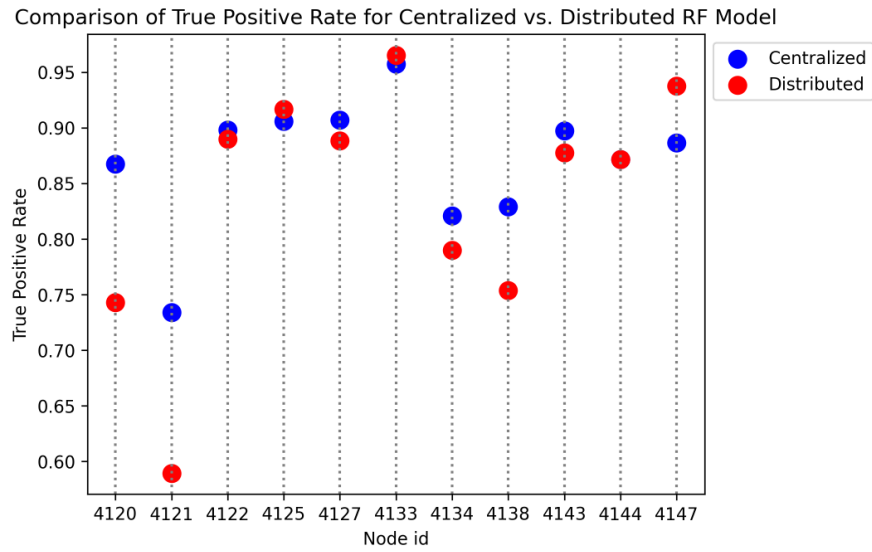
Furthermore, a direct per-node comparison between the centralized model and the distributed models was assessed. The results for each node is represented in the three scatterplots below.

Comparing the F1-score per node, the centralized model outperforms all nodes except node 4133. For eight of the eleven nodes, there is a significant increase in F1 score for the centralized model, while the remaining three nodes have very similar performance. The centralized model had an average increase in F1-score of 0.1263 across all nodes.

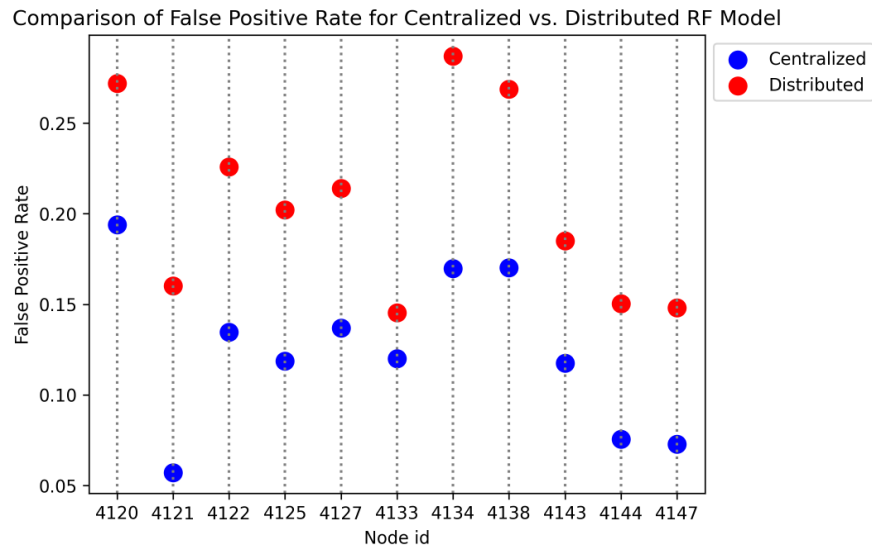


**Figure 4.9:** Scatterplot Showing F1-score per Node with Centralized and Distributed Model.

In plot 4.10b, it can be observed the FP-rate for all nodes is improved when using the centralized model. For TP-rate, the same observation cannot be made as just seven out of the eleven nodes experience improved TP-rate when using the centralized model.



(a) Scatterplot Showing TP-rate per Node with Centralized and Distributed Model.



(b) Scatterplot Showing FP-rate per Node with Centralized and Distributed Model.

**Figure 4.10:** Scatter Plots Displaying TP-Rate and FP-Rate.

## 4.4 Understanding Model Predictions

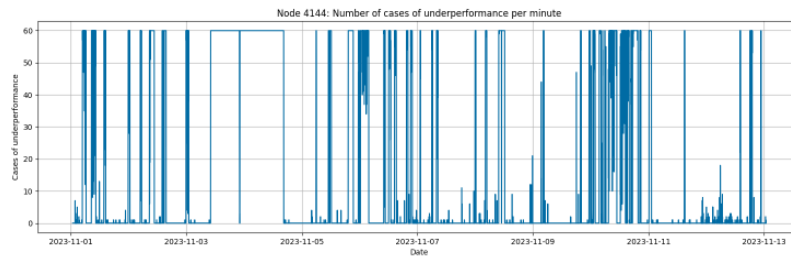
This section will present a series of diagrams in order to understand the predictions of the model. Both the centralized model and distributed models will be looked at.



### 4.4.1 General Findings

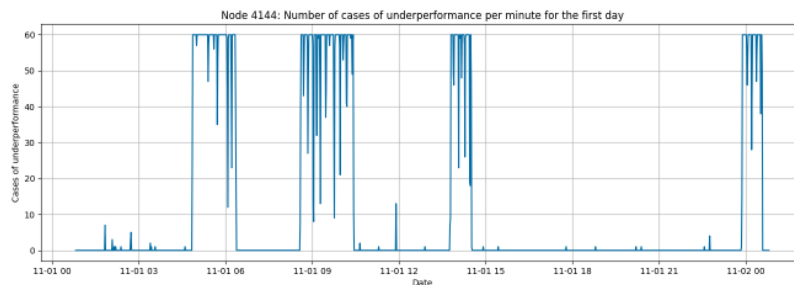
The first aspect looked at is how underperformance permittes over time. To accomplish this, a distributed model will be investigated at to get an understanding of how the performance of a node can change over time, instead of the entire network.

Figure 4.11 displays the total number of underperformance cases per minute. A continuous and long lasting section of almost total underperformance can be observed about one-fifth through the diagram. Elsewhere, there can be observed smaller periods where underperformance spikes to a maximum of 60 cases per minute. Between underperformance spikes, the number of underperformance cases per minute tends to fluctuate around zero and five.



**Figure 4.11:** Number of Underperformance Cases per Minute for the Entire Three Month Period. You may need to zoom in to see the figure text.

Figure 4.12 is the same as 4.11, only focused on the first day of the data. Here, four distinct sections of spikes in underperformance can be observed. Notice, the transition before and after a spike is not smooth, but rather very sharp. Again, observe the number of underperformance cases between spikes is close to zero.



**Figure 4.12:** Number of Underperformance Cases per Minute for the First Day.

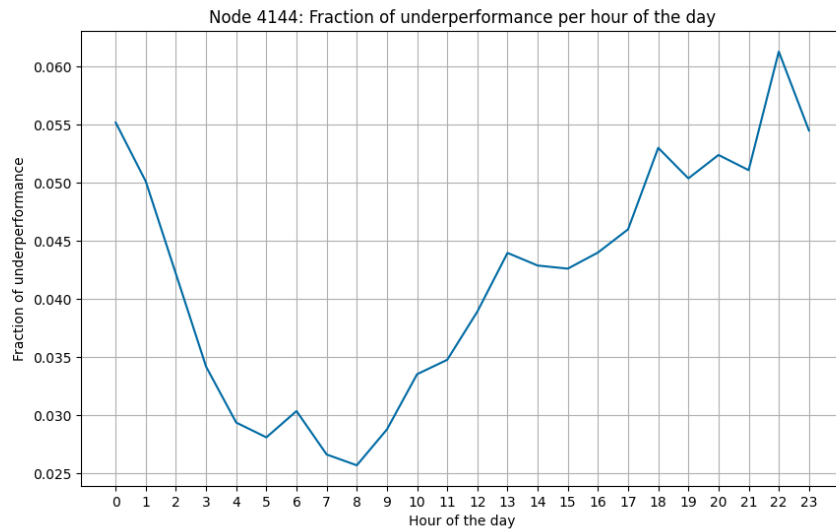
### **Length of Underperformance**

Using the same distributed model, once underperformance is predicted, the average duration of consecutive predictions of underperformance is 4.8. Since the model is based on granularity of one seconds, this equates to a 4.8 second average length of consecutive predictions of underperformance. For cases of not underperformance, the average length of consecutive cases is 10.1 seconds.

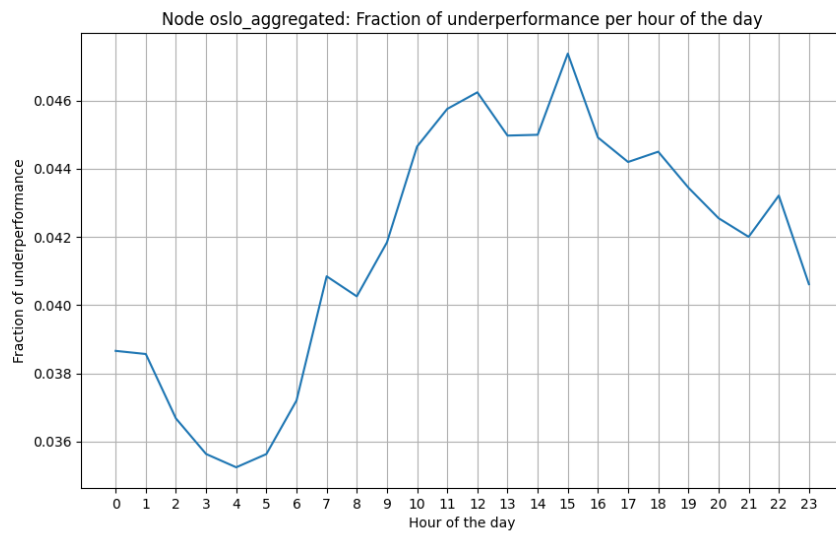
### **Underperformance Rates**

Figure 4.13 displays the fraction of underperformance for each hour of the day for the chosen distributed model and for the centralized model. This is calculated based on the entire three month period of the Centralized Dataset.

For the centralized model, it can be observed that the early hours of the day has the smallest share of underperformance. This continually increases from around 4 AM to about 4 PM where it peaks. After that, the fraction of underperformance decreases. While the distributed node also starts with a small share of underperformance in the early hours, this gradually increases and does not stop until 11 pm, peaking at 10 pm.



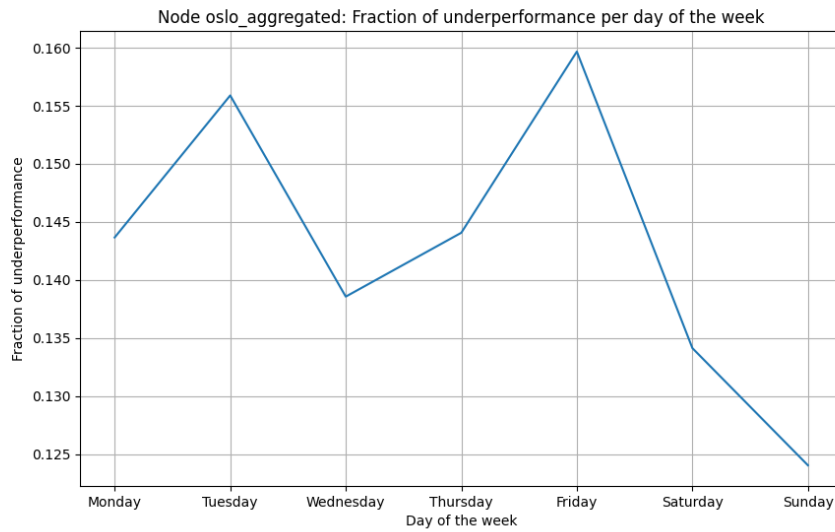
**(a)** Fraction of Underperformance per Hour of the Day for Node 4144.



**(b)** Fraction of Underperformance per Hour of the Day for the Entire Centralized Dataset.

**Figure 4.13:** Fraction of Underperformance per Hour of the Day.

Figure 4.14 shows Friday has the highest fraction of underperformance with Sunday having the least. Although, the difference between days is not large with Friday only having about 3.5% points more underperformance than Sunday.



**Figure 4.14:** Fraction of Underperformance per Day of the Week for the Entire Centralized Dataset.

#### 4.4.2 Investigating Mispredictions

This subsection will look deeper into the causes of model mispredictions. Emphasis will be put on understanding the FN as this class of misprediction is the most important as detailed in 3.5.3.

Table 4.8 displays the mean, bottom 25% and top 25% of the features received through passive measurements based on the predicted class. The centralized model will now be examined.

For RSSI, it can be observed that for the FN class, the values tend to be smaller. The same can be observed for RSRP. For RSRQ, the opposite is true for the mean and bottom 25%, where the TP have the smallest values.

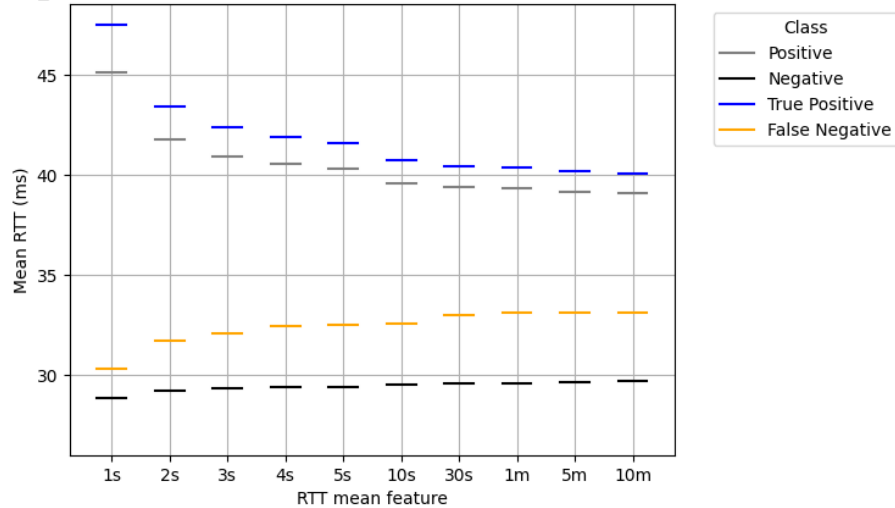
Class	Metric	Mean	Bottom 25%	Top 25%
TP	RSSI (dBm)	-56.1	-60.0	-49.0
FN		-57.4	-62.0	-51.0
TP	RSRQ (dBm)	-11.9	-14.0	-10.0
FN		-11.7	-13.0	-10.0
TP	RSRP (dBm)	-85.1	-93.0	-75.0
FN		-86.2	-95.0	-77.0

**Table 4.8:** Passive Measurement Features Categorized into TP and FN Classes.

Similarly to 4.8, Figure 4.15 displays the mean values for the actual positive, actual negative, TP, and FP classes for every RTT feature. This diagram

shows that the mean RTT values for the positive and negative classes are very far apart. Furthermore, the diagrams shows that for FN predictions, the RTT value is significantly closer to the negative class rather than the positive class, and vice versa.

Node oslo\_aggregated: Mean RTT value for the RTT mean features for various classes



**Figure 4.15:** Mean RTT Value for all RTT Features Categorized into Various Classes.

### 4.4.3 Effect of Features

In this subsection, post-hoc explainability techniques are applied to node 4144's model, referred to as the distributed model, to attain feature importances. First, the global feature importance is calculated for the distributed model using MDI, PaP, and SHAP. Thereafter, the effect of features on outputs is presented through SHAP local feature importance.

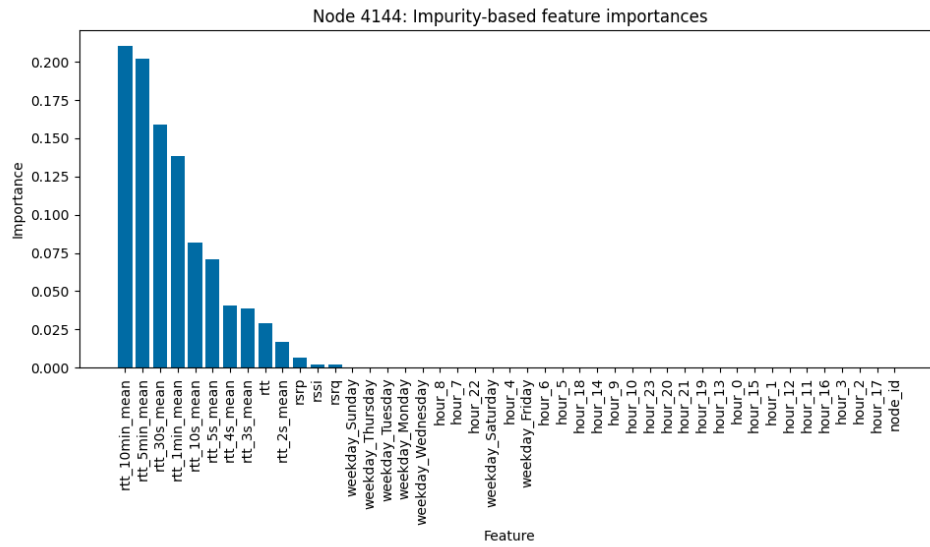
#### Global Feature Importance

One note worth mentioning is that from the global SHAP and MDI computations for the centralized model, RTT was the most important feature, valued more than double the second most important. Both of these feature importance techniques had the following top four most important features, from highest to lowest: rtt, rtt\_10min\_mean, rtt\_5min\_mean, and rtt\_1min\_mean.

In the following three diagrams, the global feature importance for the distributed model is presented.

From the MDI, figure 4.16 shows that rtt\_10min\_mean is the most import-

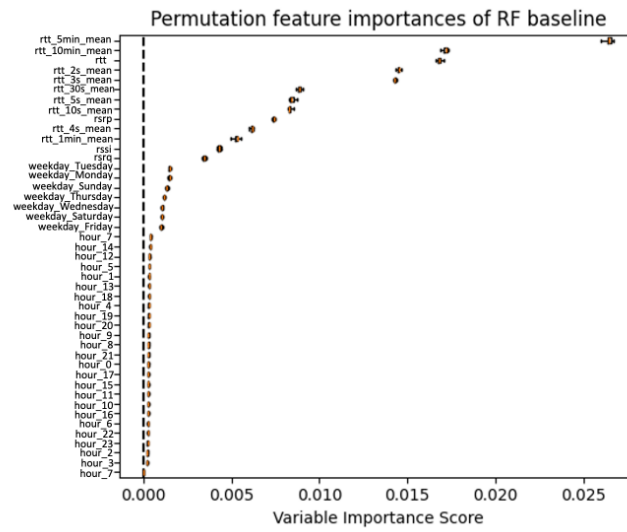
ant feature, followed closely by `rtt_5min_mean`. Furthermore, all the top 10 most important features are historical RTT features, followed by RSRP, RSSI, and RSRQ. The categorical features of time and day had seemingly no influence, according to MDI.



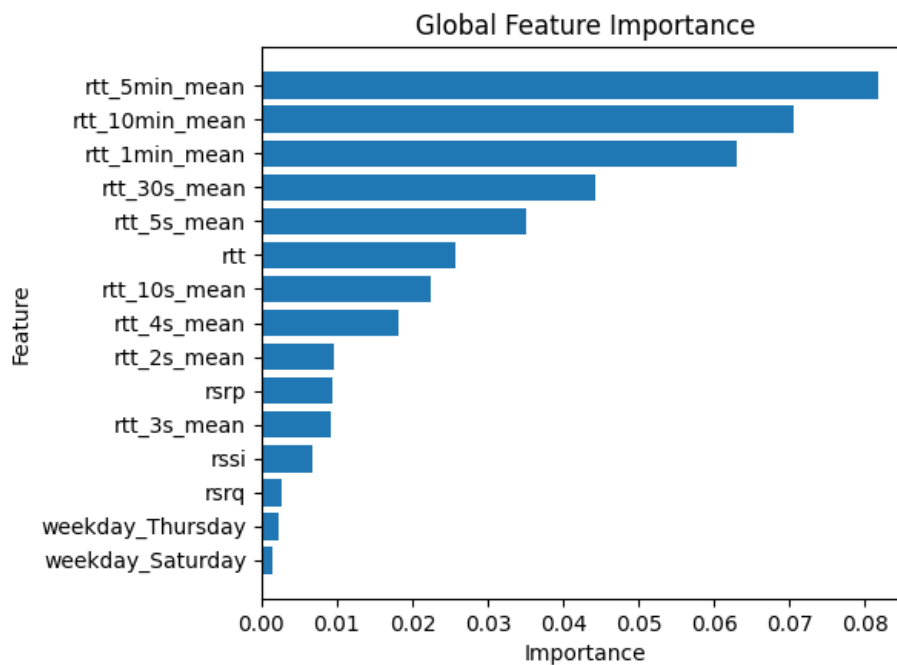
**Figure 4.16:** Feature Importance of the Model for Node 4144 using MDI.

Moreover, in Figure 4.17, the feature importance is determined through PaP. Here, the `rtt_5min_mean` feature had the highest feature importance, followed by `rtt_10min_mean` and `rtt`.

In Figure 4.18, the global feature importance is produced using SHAP. The top 15 most important features are displayed for all SHAP plots to increase the readability of the diagrams. Beyond these 15 features, the feature importance scores are barely distinguishable amongst the remaining features. Contrary to MDI, the SHAP values found `rtt_5min_mean` to be the most influential feature, followed by `rtt_10min_mean`. Unlike in MDI, RSRP has more of an impact than a historical RTT feature (`rtt_3s_mean`); however, this difference is negligible.



**Figure 4.17:** Feature Importance of the Model for Node 4144 using PaP.



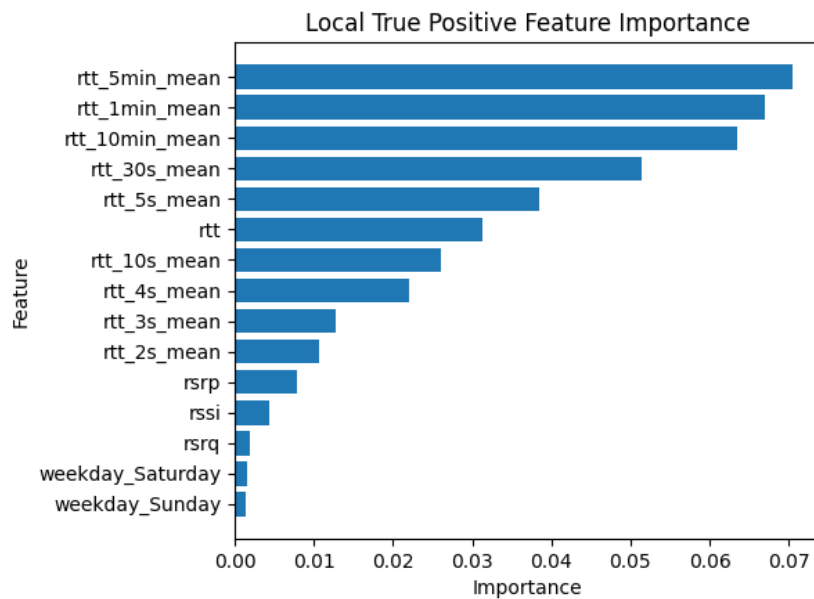
**Figure 4.18:** SHAP Global Feature Importance of the Model for Node 4144.

### Local Feature Importance

When investigating local interpretability, the local feature importance for the outputs of the distributed model was determined. Here, the model's output is split into TP, FP, TN, and FN predictions. The local SHAP values for these has then been plotted in the four bar plots seen below.

While running the distributed models, it was found that on average local interpretation on one entry with SHAP took 11.55 seconds; this average was taken over seven different attempts, with little variation between times. On the other hand, due to the size of the centralized model and the corresponding SHAP TreeExplainer, interpreting one entry took over 14 minutes.

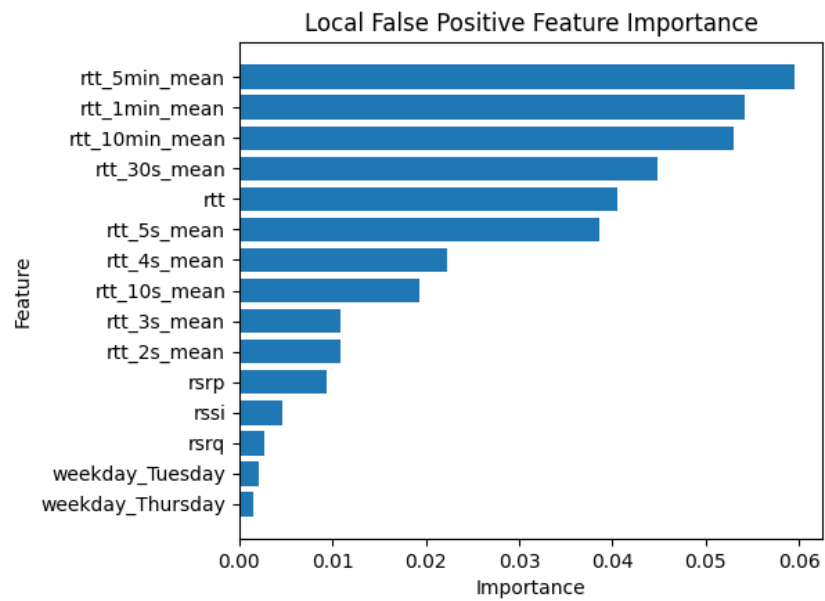
In Figures 4.19 and 4.20, the feature importance is calculated for the positive predictions of the distributed model. Here, the two plots have the same top 4 features: rtt\_5min\_mean, rtt\_1min\_mean, rtt\_10min\_mean, and rtt\_30s\_mean. Additionally, the SHAP values for these features have the same magnitude, varying only by 0.01-0.02. Moreover, the passive quality measurements (RSRP, RSSI, and RSRQ) all have a SHAP value of less than 0.01 for both plots.



**Figure 4.19:** SHAP Local of True Positive Outputs for the Distributed Model.

The two Figures 4.21 and 4.22 present the local feature importance for the negative output class. Again, the plots are quite similar to one another. In these plots, the top 5 features are respectively: rtt\_5min\_mean,

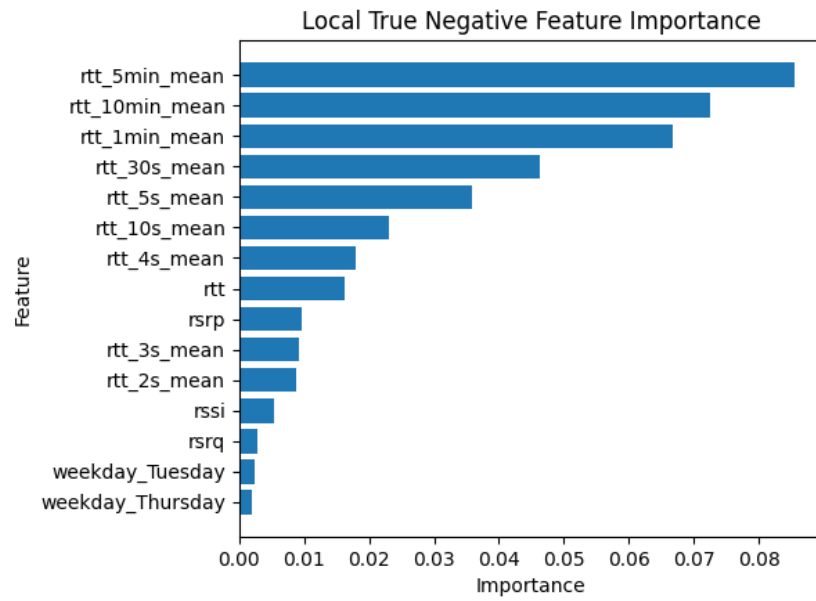




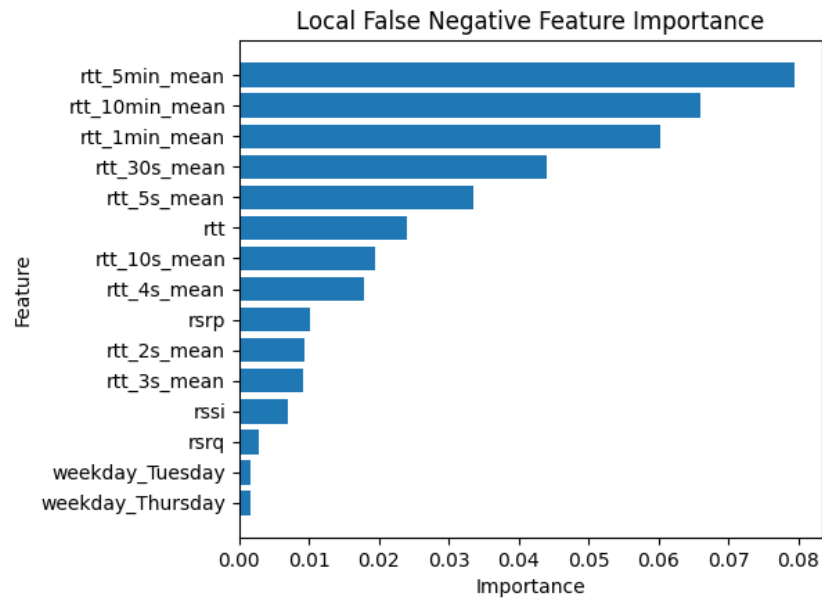
**Figure 4.20:** SHAP Local of False Positive Outputs for the Distributed Model.

rtt\_10min\_mean, rtt\_1min\_mean, rtt\_30s\_mean, and rtt\_5s\_mean. It can be seen that in both plots the distribution of importance is alike. The top 5 features make up a SHAP score of around 0.5.

Results for the centralized model was not included due time and resource limitations.



**Figure 4.21:** SHAP Local of True Negative Outputs for the Distributed Model.



**Figure 4.22:** SHAP Local of False Negative Outputs for the Distributed Model.

## 5. Discussion

In this chapter, the results from the experiments will be analyzed, interpreted and discussed. Moreover, the choices taken throughout the research process will be dissected in order to evaluate the validity of the research.

### 5.1 Comparing the ML Model Types

This section will examine the results attained during the preliminary experiment comparing performance for various ML models. Recall that the dataset used for this experiment was the Centralized Dataset.

#### 5.1.1 Choice of Metrics

As extensively detailed in 2.3.2.2, choosing the right set of validation metrics for an imbalanced dataset is difficult. Since the purpose of the research is to benefit both the end-user and the operator, the validation metric that best balances these desires was used. In this case, operators want to catch as many errors as possible, but not treat every single measurement as underperformance; therefore, an operator wants to maximize both recall but even more so precision. From the user's perspective, it is most important for their perceived underperformance to be caught. Hence, the recall metric is worth maximizing. The validation metric that attempts to find the optimal balance between recall and precision is the F1-score. The same rationale behind using the F1-score will be used throughout the experiments.

In addition to the F1-score, additional metrics such as AUC, AUPRC, and FP-rate were utilized to ensure a rounded model. Other factors than validation metrics were also considered. Feasibility of training, storage and running predictions are important considerations due to the hardware limitations on the locations the model will potentially run. This is why disk-space, training time and compute per prediction were assessed for each model.

#### 5.1.2 On Grid-Search

Due to the large hyperparameter space, running the grid-search for some of the models took from a few hours up to around five days on the HPC.

This intensive processing time comes from the combinatoric explosion of the grid-search. In hindsight, the team should therefore probably have used a randomized search, which samples the hyperparameter spaces randomly in order to approximate best parameters. This choice would have lead to drastically shorter training time, while allowing for a larger hyperparameter space to be sampled (Bergstra & Bengio, 2012). In short, this means the best model found might not be the best model possible, but rather the best model that could be found given the hardware and time restraints.

### 5.1.3 Analyzing Model Performance

Since the scoring function used in the grid-search was the F1-score, it is perhaps interesting to note that all the models tend to value recall far more than precision (see Table 4.2). In fact, the average recall across every model was 0.8710, while the average precision was 0.3746. This suggests that the models are good at predicting underperformance, but at the cost of many false predictions of underperformance (i.e FPs).

Looking at the FP-rates from the same table, observe that they tend to be low across all models (other than logistic regression which is very high). A low FP-rate may seem counterintuitive as the relatively low precision values across the board suggests the models predict many FPs in relation to TPs. One possible explanation of this discrepancy is the imbalance of the dataset. This situation appear similar to the example given in 2.3.2.2.

Another interesting discovery is that there were only a few considerable differences in the performance between models, apart from logistic regression. The logistic regression classifier had a significantly higher recall than all other models. Simultaneously, its precision was noticeably worse than the rest. This result may be an indication of the model being underfitted as it tends to classify the majority of entries as underperformance. This reasoning is further supported by the fact the logistic regression classifier has a noticeably higher FP-rate and lower FN-rate.

Despite concerns of being underfitted, from the point of view of the end-user, a model such as logistic regression may be desirable due to its effectiveness of capturing cases of underperformance. However, in a practical situation, an operator using such a model may be overloaded with FP cases, making such a model impractical from this perspective.

Looking at Figure 4.1b, the other models had drastically less variation in performance. Still, some models such as CatBoost and RF performed better than the remaining models, as seen in Table 4.2. These models

have a higher F1-score, AUC, and AUPRC compared to the other models. As such, both operators and end-users would benefit from these models.

From a practical point of view, it is important to consider the training time, prediction speed, and disk space occupied. Comparing the best performing models (CatBoost and RF), notice that CatBoost predicts with an order of magnitude faster and takes over 800 times less disk space. On the other hand, it takes 80% longer to train, although this is only 707 seconds compared to RF's 390 seconds. Whether this is significant or not is wholly dependent on contextual factors surrounding implementation.

As mentioned in 3.4.10, SVC was initially considered, but had to be removed due to its extensive training time. This meant no results for SVC were recorded. This would likely exclude SVC as a viable option for practical applications since models periodically require retraining.

#### **5.1.4 Chosen Model**

As shown in the results section 4.1.2 and discussed in 5.1.3, the best model in almost all cases was CatBoost. It is natural to question why this model was not chosen moving forward. The simple answer is that the final results for this model came too late. In the first iteration of the grid-search, poor parameter choices led to CatBoost performing worse than RF, which was best at that point in time. Towards the end of the project, the grid-search parameters for CatBoost were updated and rerun, producing much better results. However, there was not enough time to rerun all the other experiments with the new parameters. At the same time, using the absolute best model does not make significant differences for the purpose of running the experiments.

The original choice of RF was not only based on the performance results. When reading literature on earlier research, many papers used RF as a baseline model. Further, an added benefit of using RF is that it is easier to explain than boosting algorithms.

Another point that should be addressed is the choice of models to compare. No deep learning or neural networks ended up being used in the comparison. The reasoning for this choice is mainly two-fold. Most importantly, these types of models are harder to explain than traditional tree-based ML models (Robnik-Šikonja & Kononenko, 2008). As explainability of the final model is a requirement for an implementation with existing systems, this makes justifying the usage of these models hard unless they significantly outperform tree-based methods. Comparative studies, such as Borisov et al., 2021, show that deep learning methods in fact often perform worse

than the traditional models on tabular data.

### 5.1.5 Evaluating the Sampling Methods

Synthetically sampling the data took longer than expected. In the sampling experiments, an array of sampling methods were performed and tested on the best RF model. As mentioned in 3.4.9, this experiment was run on a smaller two-week subset of the Centralized Dataset. This was done to save time as running on the entire dataset took too long.

Undersampling appears to be the best sampling method based on prior research papers such as the paper Ahmed et al., 2021. This paper showed NearMiss had been used with good results on similar data. Our findings in 4.1.2 contradict this statement. Through further investigation, it was found that the mentioned paper had used sampling methods on both the train and test set of the data. This "fixes" the problem of imbalance in the test set which results in the precision metric being significantly stronger. However, by doing this, the testing set is no longer representative of real-world scenarios, meaning the improved precision value is artificially high. To test this claim, the team undersampled the testing set to achieve class parity, and recalculated the validation metrics. After doing this, the precision value significantly increased as postulated.

Oversampling methods were also tested, with the measured performance in F1-scores being promising. Nonetheless, these methods did not harbor any great improvements over random undersampling. As seen in the Table 4.4, these methods had on average better precision and F1-scores. This came at the cost of a lower recall, meaning that the model was worse at predicting underperformance. Further, as seen in the same table, the time to perform both sampling and to train the model on the sampled data was almost tenfold of random undersampling.

In addition, when trying to perform oversampling on larger datasets, the memory required became too significant. This caused HPC-jobs with 200GB of RAM to run out of memory, rendering these methods infeasible for the main experiments. Based on these tests, it was therefore chosen that random undersampling was the best sampling method in our case, having the added benefit of not relying on synthetic data. At the same time, it is worth mentioning that there are no results on how well oversampling methods perform on the full Centralized Datasets.

### Splitting the Dataset

One consideration not taken was stratification of the train-test split of the initial data. The 80-20 split of respective train and test data may result

in even further imbalancing the data. Although resampling was used, the split created had the possibility of losing distinguishing information as the proportions may be different in each of the datasets. Still, due to the large dataset and through manual validation of the deterministic split, it was confirmed that the datasets retained the same class balance post-split.

## 5.2 Time Granular Measurements

In this section, the experiments relating to time granularity will be discussed.

Given that the data is already captured by NNE every second, it may seem strange to investigate lower granularities. Additionally, the UDP packets sent by the NNE nodes to capture these measurements are only 20 bytes per packet. Still, researching methods to reduce network traffic overhead will be essential for the future with 5G and 6G implementations and increased user-devices in cellular networks (*Global Mobile Trends 2024*, 2024).

Decreasing overhead is not the only factor that motivates looking into time granularity. A combined model could be created that not only looks at the next second, but also for each subsequent granularity.

Recall that a model trained on a  $X$ -second granularity will make underperformance predictions  $X$  seconds ahead of time. If a model could accurately predict the performance at each of the next, say 60 seconds, the network operator has a better prospect of their service compared to only being able to predict 1-second ahead of time.

### 5.2.1 Interpreting Results for Varying Granularities

Looking at the results from 4.2, it is evident that the higher time granularities perform the best. Specifically, the 1-second granularity has the highest F1-score and AUPRC, although surprisingly not by much. Interestingly, the 5-second granularity actually outperforms the 1-second granularity in AUC and specificity. Further, all the granularities from 1- to 10-seconds have very similar values.

The last finding is interesting and surprising because it implies that there is no observable difference if measurements are taken every second or every tenth second. Further substantiated by the fact that the 5-second granularity outperforms the 1-second granularity in some remarks. Further research should look into how a centralized model on a larger dataset

performs on varying granularities as this could help find the optimal trade-off between overhead and performance for predicting underperformance.

The Figures in 4.2 and the Table 4.5 display a trend that performance drops off, although miniscule, as the granularities reach 30-seconds and lower. Further, the lowest granularity tested, 60-seconds, still maintains similar performance to the highest of 1-second. This indicates trends in network underperformance are detectable over longer time periods. Although, as will be discussed in 5.6.2, this might be a byproduct of how the data was split. Further, seeing that the 60-second granularity has only a slight drop off makes looking into even lower time granularities interesting.

### **Result Implications**

Recall from 3.4.8 that for the 1-second interval of measurements, the model will predict one second in advance. When decreasing the granularity, the predict interval will increase by the same amount. So for a granularity of 60-seconds, the model predicts exactly 60 seconds in advance.

As mentioned, the results for all the time granularities are quite similar. This implies that there is only a slight drop off in predictive ability when time granularities decrease. Further, this means that a future model could be set up in a manner where a model predicts not only the underperformance on the next measurement with the current granularity, but the next set of measurement granularities. In other terms, a model could predict each single second, up to  $X$ -seconds into the future, where  $X$  is the lowest granularity with satisfying performance.

Currently, the highest granularity of the active measurements is run every second. This frequency of the measurements restricts the predictive abilities of the models to 1-second granularity or lower; in other words, the model is not able to predict underperformance events occurring between one-second measurements. On the other hand, if these periods of underperformance in between two one-second measurements do not persist or recur over a longer time period, then they might not even be interesting, as mentioned by Telenor in the interview.

### **60 Seconds and Beyond**

The team wanted to gain an as holistic view as possible when choosing the granularities for this experiment. While looking through research, there was a lack of experimentation on changes in granularity. Therefore, the choice of the time granularities was based off of the team's experience with similar problems. It turned out that the lowest granularity of 60-seconds



ended up performing well, suggesting even lower granularities should have been considered.

## 5.3 Centralized vs. Distributed Models

This section will examine the results attained during the experiments comparing a centralized model vs. distributed models in section 4.3.

Prior to these experiments, it was unknown whether a centralized model, trained on a larger dataset, may lead to better performance. Therefore, the motivation for this experiment was to understand the performance difference between a centralized model and distributed models.

### 5.3.1 Interpreting the Results

Looking at the results, it is clear that the centralized model outperforms the distributed models. Based on these results, the most direct advantages of a centralized model would therefore be to capture more instances of underperformance, while minimizing the workload for operators in regards to FP. This may suggest that having more quantity and variance in data from multiple nodes enabled the centralized model to better understand underperformance compared to distributed models only trained on one node.

At the same time, other factors may influence this increase in performance which alter what the model actually predicts. When discussing the differences, it is vital to remember the threshold for underperformance in the centralized model is the same for all node data, i.e. set as top 10% cut-off of RTT for all nodes. The nodes that have an average lower RTT may therefore be easier to distinguish with regards to underperformance.

Recall also that for the distributed models, the threshold values are calculated based only on the single node considered for the respective distributed model. This means that the centralized and distributed models may have different definitions of what constitutes as underperformance, even when looking at the same nodes.

From the point of a telecommunication operator wanting to apply this to self-healing, the distributed models may initially make more sense as it gives better insight into how each individual node performs. However, for the centralized model, every single node experience decreased FP-rate compared to the distributed models. For practical implementations, FP-rate is important for telecommunication operators as too many FPs would overload the operator.

From the Telenor interview, it was made apparent that providing the best average user experience is important. The centralized model exhibits less spread in performance across each node, meaning acting on the under-performance predictions would be more evenly spread using this model.

### **5.3.2 Benefits of a Centralized Model**

From the perspective of an SLA, a centralized model makes the most sense. As the SLA does not account for a single location's performance, using a generalized threshold for all nodes in a geographical area therefore seems the most sensible. The SLA does not provide any guarantees on the performance of the end-user device, but rather on the delivered performance from the network. An upperbound of acceptable latency is instead described in the contract, among other metrics. With 5G and beyond, there is a considerable focus on low latency and ultra high reliability in the network, making this an intriguing path going forward.

Another potential benefit with a centralized model is the ease of introducing new nodes and the ability to understand changes in the network. Using previous training data from the other nodes, the centralized model may be able to transfer its understanding to predicting the new node. Furthermore, the concept of transferability from one operator to another was tackled in Ahmed et al., 2021 and indicates the possibility of using one model for multiple operators. Again, more research here is necessary for any conclusions to be made.

### **5.3.3 Benefits of Distributed Models**

In Ahmed et al., 2021, the authors discover the need for retraining due to model degradation occurring as time progresses more than a week beyond the end of the training data. It is important to note that the aforementioned paper looks at a centralized model based on three nodes and therefore this trend may have a different affect on a larger centralized model and on the distributed models. Further research on this topic will be necessary to draw conclusions.

In any case, one of the considerations in choosing an ML model is the training time. As the results in Table 4.6 show, the training time for the centralized model increased accordingly with the dataset, meaning each distributed model needed less training time.

## 5.4 Understanding Model Predictions

This section will discuss the findings from the exploration of the model outputs based on the results presented in 4.4. Understanding the model predictions is important as this can be used to understand the trends and patterns captured by the model. Furthermore, such an understanding enables identification of areas and situations where the model is proficient and where the model falls short.

### 5.4.1 Trends in Underperformance

Looking at underperformance temporally from Figures 4.11 and 4.12, a few interesting observations can be made. Firstly, once underperformance occurs, it is severe, in this case meaning large portions of entire minutes are filled with underperformance. Secondly, these sections of underperformance tend to start and end abruptly. There is no smooth curve going from little underperformance to a lot of underperformance. Instead, very sharp, often 90 degree bends, can be observed from the two diagrams.

These two observations suggest underperformance occurs suddenly, and that once it occurs it becomes all-engrossing for its duration. On the other hand, between the spikes of underperformance, performance is generally acceptable without much deviation.

Looking at the fraction of underperformance per hour of the day in Figure 4.13, the expected daily pattern emerges. During the late night and early morning, underperformance is low as network traffic is low. However, it is difficult to say whether this is due to less congestion because of low traffic or some other underlying condition.

The fraction of underperformance per day of the week also shows an interesting pattern. During Saturdays and Sundays, the fraction of underperformance is less than the week-days. Any inference of the causes would be pure speculation and should be researched further on its own.

### 5.4.2 Causes of Mispredictions

Looking at positive cases that the model mistakenly predicts as negatives, FN, provides insight into the criteria the model prioritizes when classifying an instance. Nonetheless, attempting to discern patterns and correlations in this manner proves only insights and is not conclusive.

Table 4.8 shows there is a non-trivial difference in the mean values in RSSI, RSRQ and RSRP between the TP and FN classes. This suggests that the values of these features on average influence model misprediction.

For RSSI and RSRP, the FN class has a lower mean value than the TP class signifying that the model may wrongly attribute non-underperformance in these cases. Interestingly, RSRQ has a higher mean value in FN cases than the TP class. This coincides with the reading of the data in Figure 3.15, which shows that RSRQ has a negative correlation with the aforementioned features.

Examining the mean values for all RTT features per class in Figure 4.15, a much clearer pattern emerges. A distinct delta in mean RTT can be observed between the positive and negative classes. Furthermore, for the FN class, the mean RTT is very close to the mean RTT value for the negative class, while quite far away from the positive class. At the same time, the mean RTT value for the TP class is very high. This suggests that the model prioritizes the RTT value a great deal, and that a low RTT value for an actual positive is a good indicator that the model may mispredict.

This insight also implies that looking solely at the most recent RTT measurement does not provide enough information to correctly predict all instances of underperformance.

### 5.4.3 Explainability of Model Predictions

After the interview with Telenor, the team confirmed the necessity for explainability in the ML model. Additionally, more international legislation, such as GDPR and EU's AI Act, is requiring a certain degree of explainability in critical systems, where telecommunication is defined as critical infrastructure (European Parliament, 2023). Explainability in ML systems is consequently crucial for future integration.

Still, the degree to which explainability is necessary is imprecise and dependent on the use-case. From the operator's point-of-view, it is important to understand both the reasons for the conclusion drawn from the model but also the logical mechanisms of the model. On the other hand, an end-user may be satisfied with an explanation of why their underperformance did not raise an alarm. To this end, experiments were conducted to better understand both the innerworkings and outputs of the model.

Since the chosen model, RF, is a gray-box model, it lacks intrinsic interpretability. Some may still argue that the ensemble nature of the model could allow an operator to look at individual learners; however, practically, this approach will be infeasible due to the sheer number of learners. The task of explaining the inner-workings of the model was therefore investigated through looking at the model's tendencies in 5.4.2. Through understanding the patterns of the model, the operator has more of a basis

for determining the true nature of a prediction, how likely it is to be true or false. This understanding also acts as a basis for instilling trust between the operator and the model.

Furthermore, the team ran multiple post-hoc explainability techniques on both the centralized model and node 4144's distributed model to gain a better understanding of the model's choices. It is important to remember that although a model must be explainable, this explainability must be practical. During experimentation with SHAP global feature importance for the centralized model, the program ran for over four days without giving any results. A general understanding of larger models may therefore be restricted based on time restraints. For this reason, explainability on the centralized model was stopped after this attempt.

Still, for the overall understanding of the model, global feature importance is useful. The various feature importance techniques utilized during the investigation of global model output provide different advantages and disadvantages when it comes to evaluating the influence of features. Both MDI and PaP are susceptible to optimistic portrayal of linearly correlated features (Greenwell & Boehmke, 2020)(Hooker et al., 2021). Furthermore, in both of these methods, the historical RTT data, which was strongly positively correlated with one another, were nearly all of the most important features. Moreover, SHAP may experience similar problems of unrealistic explanations when the features are dependent on one another (Aas et al., 2021). The validity of the feature importance must therefore be further evaluated before trusting.

However, from a broader point of view, it is still interesting to look at the possibility of locally interpretable results. Local interpretation would be useful for both end-users and operators, allowing for single predictions to be explained. Furthermore, with the single entry local interpretation, it was shown the time to calculate feature importance for smaller models was relatively small. This signifies the higher usability of local interpretation in practical applications.

Moreover, in the experiment conducted, the two output classes (positive and negative) were compared. As Rudin, 2019 discusses, it is important to investigate both the correct and incorrect classes when explaining a model. From the results in 4.4.3, the distributed model shows a tendency to value the same features for a given output class when mispredicting, with the same top four to five features. Again, since these values are all historical rtt features, care has to be taken when interpreting the results. One implication of these results, when compared with the results from Figure 4.15, is that the model heavily relies on the historical RTT data when

making a decision. Therefore, when the historical RTT values lay closer to the mean value, it becomes more difficult to determine the output class and mispredictions occur.

A useful application of the feature importances is to reduce the noise of the train data through removing unimportant or uninfluential features. Assuming the validity of the global SHAP values for the distributed model, all hour categories could be removed from the current model. Ultimately, by understanding which features the model looks at when mispredicting, an operator has a starting point for diagnostics on a prediction.

### **Result Implications**

When conducting research on model choice, focus should be placed on the explainability-performance trade-off. If a model can be made interpretable and still maintain an acceptable performance, then this may be a good starting point. Instances with more complex relationships may warrant more complex models. However, when choosing the post-hoc explainability techniques to aid in understanding, it is essential to be aware of the pitfalls associated with it. For this particular issue, SHAP has been shown to provide both local and global interpretation with relatively few downfalls.

An important consideration in using post-hoc explainability techniques is their restrictions. According to Fisher et al., 2019, feature importance determined from one model may not be transferable to another model; this may indicate that the explanations produced for one model is not necessarily transferable even within the same model type, meaning new explanations may have to be generated for each re-training.

## **5.5 Supplementing Self-Healing Systems with End-User Data**

In current literature, there is a lack of discussion on how end-user data can be integrated into self-healing systems; nearly all papers focusing on fault-detection utilizes real or simulated intra-network data. Determining the specific system for which this research may be incorporated is a tremendous task given the scope of this thesis and the team's competency in the domain.

Instead, a broader view of the usefulness of end-user data can be taken, highlighting how this data can enhance the understanding of user-experience and usage as an early warning system. ML models based on end-user data can be used through both centralized and distributed models predict-

ing underperformance ahead of time. This may give telecommunication operators a better understanding of when users will experience underperformance in their networks. With this understanding, proactive systems could be built that aim to solve underperformance before it occurs.

During the interview with Telenor, when asked the question: "Would an automatic solution based on end-user data, as described at the start of the interview, be interesting to Telenor?", the answers were that from a design point of view this is relevant and that end-user measurements could be used to verify intra-network measurements. By extension, we postulate that another approach going forward is to utilize systems based on end-user data to validate systems based on intra-network data.

In this thesis, the usage of end-user data in predictive systems was investigated. One of the main takeaways from the experiments was the numerous considerations needed. Before integration into self-healing systems becomes viable, more research must be undertaken.

## 5.6 Reflection on Methodology

This section reflects on the research methodology, focusing on reproducibility, validity, and setbacks encountered. This includes the use of end-user data, the necessity of HPC, and challenges faced during the research process.

### 5.6.1 Reproducibility of the Research

While contemporary research in the field of self-healing solutions often use operator-owned intra-network data, this paper uses more readily available end-user. With the datasets attained from the NNE platform and code made publicly available on GitHub<sup>1</sup>, this aspect of reproducibility is not a concern.

On the other hand, one aspect that reduces the ability to reproduce the results is the requirement for using a HPC. Without this resource, attaining the results would have taken substantially more time and using larger datasets may be infeasible.

### 5.6.2 Validity of Research

The data used in this research has timestamps attached and could therefore be looked at as a timeseries. This was not taken into account when splitting the data into the train and testing sets. As seen in 4.4, there is

---

<sup>1</sup><https://github.com/TrymNOHG/bachelor-thesis>

a temporal correlation of underperformance. An effect of splitting at random could be that model performance is artificially high as the model has already been trained on data from the same spike pattern. For future experiments, a test set should therefore be separated based on time.

As seen in 4.1, the final RF model had a max-depth value of None. In other words, the RF could grow trees as deep as it wanted. This is worrying as this allows the RF to capture all the complexity of the dataset in its trees, overfitting to what it already has seen. Based on the huge size of the disk space used by the model, this has also most likely happened. As mentioned, the data is split in such a manner that spike patterns could be in both the train and test split. This means that almost identical patterns could be already captured in full within the trees of the forest. The results in the explainability section also shows that the RF model greatly values current and historical RTT values, which further point in this direction. To summarize, there are some worries about the models being overfit.

### **5.6.3 Setbacks**

This section will detail the setbacks that occurred and how the team dealt with them.

#### **Labelling the Dataset**

During the initial stages of the thesis, the team was under the impression that the dataset provided by SimulaMet would come pre-labelled. The team thought the labels were set by subject matter experts – the operators – and mapped to actual intra-network faults.

This was not the case, meaning the team had to invest a non-trivial amount of time into discussing new approaches to label the dataset. The new labelling process is discussed in 3.4.8. Since the labels now are not directly mapped to a fault occurring in the network, performing fault diagnosis did not become an option anymore. Furthermore, proposed research directions had to be scrapped due to the lack of expert labels. Instead, the team worked quickly and effectively to pivot the focus of the research to more relevant areas.

#### **Research Pivot**

Based on the description for the thesis, the team was under the impression that the main purpose of the research was fault detection and root cause diagnosis within the SON framework. With this in mind, a lot of time in the beginning of the project was spent researching SON-related topics and



performing a literature review. Based on this, research questions were formulated and presented to both supervisors.

Upon access to the dataset, two things became clear. Firstly, given end-user data without labels, research into a fault detection or diagnosis systems would not be possible since there is no tether back to an actual intra-network fault. Consequently, it was necessary to pivot the direction of research. The second realization was that the process of going from raw data to a procured dataset ready for ML purposes would be a larger task than anticipated. As a result, the new research direction had to have a narrower scope.

### **Usage of HPC**

As outlined in choice of technologies, 3.3.1, a HPC had to be used to train the ML models. A consequence of this is that the team became vulnerable to any potential downtime on NTNU's HPC. In fact, this happened multiple times during the start of May where it was not possible to obtain any compute power on IDUN for longer stretches of time.

The team anticipated that this may be an issue ahead of time, and determined that if this were to happen, the team would continue the research by using a smaller subset of the datasets manageable to run without access to a HPC.



## 6. Conclusion

In this thesis, the feasibility of using ML models based on end-user data as a supplement to self-healing solutions has been investigated. This topic was explored through three experiments focusing on performance. The first research question addressed was:

**“How does modulating the granularity of active end-user measurements affect model efficacy when predicting cellular network underperformance?”**

The discussion in 5.2 reflects upon the results gathered from these experiments ran on various time granularities for active RTT measurements. It was shown that the same model trained on decreasing time granularities share very similar trends in performance. This indicates that modulating measurement interval up to 60 seconds only has a slight negative impact on model performance. Therefore, this paper concludes that predicting underperformance in cellular networks using end-user data follows trends that are predictable up to at least 60 seconds ahead of time, with some concerns about the validity of these results. Moreover, the performance difference between the 1- and 10-second intervals are minuscule.

In addition to looking into the granularity of the active end-user measurements, performance of centralized models versus distributed models were also of interest. This led to the formulation of the second research question:

**“What are the trade-offs in deploying distributed machine learning models for each NorNet Edge node versus a centralized model for predicting cellular network underperformance?”**

Based on the results and discussion 5.3, the centralized model outperforms the distributed models by a non-trivial margin, having an average increase in F1-score of 0.1263. Comparing the two, the centralized model accomplished a higher recall and precision; two metrics that are key to satisfying both operators and end-users. Furthermore, the centralized model provides an easier and potentially less data-intensive integration of new nodes. Instead of training a new node from scratch, the other nodes’ data could help in establishing trends. On the other hand, aggregating all NNE nodes into a centralized model increases training time and disk usage proportionally to the dataset. Another major benefit with a centralized model would be the logical integration of an SLA’s latency threshold.

Explainability and interpretability, according to an interview with Telenor, is essential for network operators. Furthermore, critical infrastructure under EU's AI Act 2023 requires explainability. Therefore, through experiments on the ML models, the last sub-question was explored:

**"What considerations need to be taken regarding explainability and interpretability before integration of an automatic network solution becomes viable?"**

Explainability and interpretability serve the role of instilling trust in the ML model's understanding of the problem and in the outputs. The definition of trust may, however, differ depending on the person. Operators, with subject matter expertise, may for example require a system where the internal mechanisms are explainable. From investigating the distributed model's outputs, it became clear that before an automatic network solution becomes viable, a thorough investigation must be undertaken to understand the general tendencies of the solution. If the solution is not intrinsically interpretable, then post-hoc explainability techniques such as SHAP may be necessary. However, it is essential to determine the potential downfalls of the techniques prior to use, such as time needed to run and misrepresentative values.

Finally, to address the larger goal of the research performed in this thesis, the prior questions review to a larger degree how end-user data can be used to supplement already existing self-healing systems. This led to the formulation of the final and main research question:

**"How can a predictive supervised machine learning model handle end-user data to supplement self-healing systems based on 3GPP's specifications?"**

The research in this thesis suggests that an ML model based on end-user data predicting underperformance ahead of time provides solid results. Specifically, the value this provides and how this can be integrated with existing self-healing solutions to extract such value is an open question needing further research. Ultimately, this thesis concludes ML models, based on end-user data, can effectively predict underperformance up to 60-seconds in advance, giving operators a better understanding of user experience. For this purpose, a centralized model is shown to be better than distributed models. In addition, this paper has presented how tree-based models can be explained to this extent.

## 7. Societal Impact

One of the overarching goals of this thesis was to explore ways of improving user-experience in cellular networks. As such, the thesis hopes to positively affect various areas of society. Some of the positive implications this thesis aims to aid with are listed below:

- As cellular networks continue to evolve and grow, the resilience and reliability of the infrastructure is no longer merely a technical requirement but also of critical importance to society as a whole. Through the exploration of utilizing end-user data as a supplement to self-healing systems, this thesis advances research within improvement of the reliability of cellular networks. In turn, this improves the resilience of critical infrastructure.
- New opportunities arise with increased network reliability. An example of this is healthcare delivered through telemedicine (Georgiou et al., 2021). For example, Norwegian telecommunication operator Telenor claims the new 5G technology will enable real-time remote surgery, eliminating the necessity of acquiring expertise on-site resulting in a cheaper, faster and more available service. Generally, Telenor suggests any application that involves human and machine co-operation could be done remotely over 5G, claiming wide spanning gains for both the industry and society as a whole (Telenor, 2021).
- As outlined in 1.1, telecommunication operators spend significant amounts of revenue (23% to 26%) on operational expenditure (Asghar et al., 2018). Solutions such as the one proposed in this thesis attempt to remedy some of these costs by aiding automatic self-healing solutions. Therefore, a proposed benefit for telecommunication operators is decreased operational expenditure.

However, the research may also lead to potentially negative societal implications. Listed below are two examples:

- One concern for systems based on end-user data is privacy and ethics surrounding storage of private data. This aspect is very important to keep in mind when developing systems based on such data, especially as internet data can be classified as sensitive.

- Another concern is the power usage associated with ML systems, especially as the data operated on is large. This may negatively contribute to sustainability goals. In the case of this research, the HPC used runs on green electricity produced and located in Norway; however, this type of clean electricity is not true for all countries.

## 8. Further Work

In this chapter, the team proposes areas of interest that may help in advancing this field of research. The topics chosen are not extensive; instead, a subset of potential areas are chosen based on the experience gained from the research and experiments.

One substantial area of further research is investigating how the predictive model from this research can be integrated into contemporary SON pipelines. A meta-study on self-healing by Asghar et al., 2018 also brings this up and suggests "[...] user behavior load prediction models can be generated using machine learning techniques [...] All this information will be fed to the proactive fault prediction algorithms which would sit alongside a reactive Self-healing triggering algorithm [...]" (Asghar et al., 2018). Given a reactive self-healing triggering algorithm, further research on integrating the findings from this thesis would be intriguing and necessary for any real world usage.

The granularity results were investigated to a maximum of 60 second intervals, as discussed in 5.2.1. However, the results only show a 7% drop-off in performance, with respects to the F1-score, between 1-second and 60-second granularities. As such, it would be interesting to investigate how lower granularities would perform.

An interesting idea arising from the models based on varying granularities is that of an adaptive solution for measuring underperformance. This approach involves a model trained on every time granularity up to, say ten seconds, making predictions. In this way, the system would only need measurements every ten seconds. However, on prediction of underperformance within the next ten seconds, the system adapts and starts performing higher granularity measurements to capture the state of the network. The proposition here is that this approach would minimize overhead, while still being able to capture underperformance.

During the interview with Telenor, it became clear that one-off spikes in underperformance are not as interesting as the seasonality and trends in underperformance. To capture the patterns in underperformance, timeseries forecasting is a relevant field of study. The timeseries analysis portion of this research paper and from Ahmed et al., 2021 may act as a starting point for further investigation on this topic. The purpose of the end-user data would therefore be to provide a historical view of the underperform-

ance for single nodes as well as geographical clusters of nodes.

Moreover, within Norwegian and European laws, there are considerable restrictions on the usage of end-user data. Privacy is therefore a significant task to handle. An active area of research that works on the intersection between privacy and ML is federated learning. The discussion surrounding centralized vs. distributed models could act as a basis for further research within this topic.



# Bibliography

- 3GPP. (2022). 32.541: 'self-organizing networks (son); self-healing concepts and requirements' (tech. rep.). The European Telecommunications Standards Institute.
- Aas, K., Jullum, M., & Løland, A. (2021). Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 298, 103502. <https://doi.org/https://doi.org/10.1016/j.artint.2021.103502>
- Aczel, B., Bago, B., Szollosi, A., Foldes, A., & Lukacs, B. (2015). Is it time for studying real-life debiasing? evaluation of the effectiveness of an analogical intervention technique. *Frontiers in Psychology*, 6. <https://doi.org/10.3389/fpsyg.2015.01120>
- Afroz, F., Subramanian, R., Heidary, R., Sandrasegaran, K., & Ahmed, S. (2015). Sinr, rsrp, rssi and rsrq measurements in long term evolution networks. *International Journal of Wireless & Mobile Networks*.
- Ahmed, A. H., Hicks, S., Riegler, M. A., & Elmokashfi, A. (2021). Predicting high delays in mobile broadband networks. *IEEE Access*, 9, 168999–169013. <https://doi.org/10.1109/ACCESS.2021.3138695>
- Akbari-Moghanjoughi, A., Amazonas, J., Boada, G., & Solé-Pareta, J. (2019). Service level agreements for communication networks: A survey. *INFOCOMP Journal of Computer Science*, 18, 32–56. <https://doi.org/10.48550/arXiv.2309.07272>
- Altman, N., & Krzywinski, M. (2017). Ensemble methods: Bagging and random forests. *Nature Methods*, 14, 933–934. <https://doi.org/10.1038/nmeth.4438>
- Asghar, A., Farooq, H., & Imran, A. (2018). Self-healing in emerging cellular networks: Review, challenges, and research directions. *IEEE Communications Surveys Tutorials*, 20(3), 1682–1709. <https://doi.org/10.1109/COMST.2018.2825786>
- Batista, G., Prati, R., & Monard, M.-C. (2004). A study of the behavior of several methods for balancing machine learning training data. <https://doi.org/10.1145/1007730.1007735>
- Behimehr, S., & Jamali, H. R. (2020). Cognitive biases and their effects on information behaviour of graduate students in their research projects. *JOURNAL OF INFORMATION SCIENCE THEORY AND PRACTICE*, 8(2). <https://doi.org/https://doi.org/10.1633/JISTaP.2020.8.2.2>

- Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization.
- Berrar, D. (2018, January). Cross-validation. <https://doi.org/10.1016/B978-0-12-809633-8.20349-X>
- Bogatu, A., Paton, N. W., Fernandes, A. A. A., & Koehler, M. (2018). Towards Automatic Data Format Transformations: Data Wrangling at Scale. *The Computer Journal*, 62(7), 1044–1060. <https://doi.org/10.1093/comjnl/bxy118>
- Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2021). Deep neural networks and tabular data: A survey. *CoRR*, abs/2110.01889. <https://arxiv.org/abs/2110.01889>
- Brownlee, J. (2023). Data sampling methods for imbalanced classification [Accessed: 2024-05-07]. <https://machinelearningmastery.com/data-sampling-methods-for-imbalanced-classification/>
- Chen, K.-M., Chang, T.-H., Wang, K.-C., & Lee, T.-S. (2019). Machine learning based automatic diagnosis in mobile communication networks. *IEEE Transactions on Vehicular Technology*, 68(10), 10081–10093. <https://doi.org/10.1109/TVT.2019.2933916>
- Chen, T., Zhang, H., Chen, X., & Tirkkonen, O. (2014). Softmobile: Control evolution for future heterogeneous mobile networks. *IEEE Wireless Communications*, 21(6), 70–78. <https://doi.org/10.1109/MWC.2014.7000974>
- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754. <http://arxiv.org/abs/1603.02754>
- Colpitts, A. G. B., & Petersen, B. R. (2023). Short-term multivariate kpi forecasting in rural fixed wireless lte networks. *IEEE Networking Letters*, 5(1), 11–15. <https://doi.org/10.1109/LNET.2023.3242680>
- Cook, J., & Ramadas, V. (2020). When to consult precision-recall curves. *The Stata Journal*, 20(1), 131–148. <https://doi.org/10.1177/1536867X20909693>
- Cristianini, N., & Ricci, E. (2008). Support vector machines. In M.-Y. Kao (Ed.), *Encyclopedia of algorithms* (pp. 928–932). Springer US. [https://doi.org/10.1007/978-0-387-30162-4\\_415](https://doi.org/10.1007/978-0-387-30162-4_415)
- Dorogush, A. V., Gulin, A., Gusev, G., Kazeev, N., Prokhorenkova, L. O., & Vorobev, A. (2017). Fighting biases with dynamic boosting. *CoRR*, abs/1706.09516. <http://arxiv.org/abs/1706.09516>
- Dube, L., & Verster, T. (2023). Enhancing classification performance in imbalanced datasets: A comparative analysis of machine learning models. *Data Science in Finance and Economics*, 3(4), 354–379. <https://doi.org/10.3934/DSFE.2023021>
- Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., & Ranjan, R. (2023). Explainable ai

- (xai): Core ideas, techniques, and solutions. <https://doi.org/10.1145/3561048>
- European Parliament. (2023, June). *Eu ai act: First regulation on artificial intelligence* [Accessed: 2024-05-21]. <https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence>
- European Telecommunications Standards Institute. (2019). *Telecommunication management; Self-organizing networks (SON); Concepts and requirements (3GPP TS 32.500 version 17.0.0 Release 17)* (Technical Specification No. TS 32.500) ([Online; accessed April 8, 2024]). 3rd Generation Partnership Project (3GPP). [https://www.etsi.org/deliver/etsi\\_ts/132500\\_132599/132500/17.00.00\\_60/ts\\_132500v170000p.pdf](https://www.etsi.org/deliver/etsi_ts/132500_132599/132500/17.00.00_60/ts_132500v170000p.pdf)
- Fawcett, T. (2006). An introduction to roc analysis [ROC Analysis in Pattern Recognition]. *Pattern Recognition Letters*, 27(8), 861–874. <https://doi.org/https://doi.org/10.1016/j.patrec.2005.10.010>
- Fisher, A., Rudin, C., & Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously.
- Florkowski, C. (2008). Sensitivity, specificity, receiver-operating characteristic (roc) curves and likelihood ratios: Communicating the performance of diagnostic tests. *Clin Biochem Rev*.
- Foster Provost, T. F. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. *Proc. Third Internat. Conf. on Knowledge Discovery and Data Mining (KDD-97)*.
- Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119–139. <https://doi.org/https://doi.org/10.1006/jcss.1997.1504>
- Fürber, C. (2016, January). *Data quality management with semantic technologies*. <https://doi.org/10.1007/978-3-658-12225-6>
- Georgiou, K. E., Georgiou, E., & Satava, R. M. (2021). 5g use in healthcare: The future is present. *JSLs : Journal of the Society of Laparoendoscopic Surgeons*, 25(4), e2021.00064. <https://doi.org/10.4293/JSLs.2021.00064>
- Global mobile trends 2024* (tech. rep.). (2024, February). GSMA Intelligence. London, UK. <https://www.gsmainelligence.com>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [<http://www.deeplearningbook.org>]. MIT Press.

- Greenwell, B. M., & Boehmke, B. C. (2020). Variable Importance Plots—An Introduction to the vip Package. *The R Journal*, 12(1), 354. <https://doi.org/10.32614/RJ-2020-013>
- Hämäläinen, S., Sanneck, H., & Sartori, C. (2011). *Lte self-organising networks (son): Network management automation for operational efficiency*. Wiley. <https://books.google.no/books?id=vM2gUROYpNIC>
- Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.
- Heuer, R. J. (1999). *Psychology of intelligence analysis*. Center for the Study of Intelligence.
- Hevner, A. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19.
- Hooker, G., Mentch, L., & Zhou, S. (2021). Unrestricted permutation forces extrapolation: Variable importance requires at least one more model, or there is no free variable importance.
- Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 3149–3157.
- Khalil, A. M., Mahbooba, B., Timilsina, M., Sahal, R., & Serrano, M. (2021). Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model. *Complexity*, 2021, 6634811. <https://doi.org/10.1155/2021/6634811>
- Kirkwood, B. R., & Sterne, J. A. (2010). *Essential medical statistics*. John Wiley & Sons.
- Kurose, J., & Ross, K. (2013). *Computer networking: A top-down approach: International edition*. Pearson Education Limited. <https://books.google.no/books?id=IKCpBwAAQBAJ>
- Kvalbein, A., Baltrūnas, D., Evensen, K., Xiang, J., Elmokashfi, A., & Ferlin-Oliveira, S. (2014). The nor-net edge platform for mobile broadband measurements [Special issue on Future Internet Testbeds – Part I]. *Computer Networks*, 61, 88–101. <https://doi.org/10.1016/j.bjp.2013.12.036>
- LaValley, M. P. (2008). Logistic regression.
- Lazar, R., Militaru, A.-V., Caruntu, C., Pascal, C., & Patachia Sultanoiu, C. (2023). Real-time data measurement methodology to evaluate the

- 5g network performance indicators. *IEEE Access*, PP, 1–1. <https://doi.org/10.1109/ACCESS.2023.3271366>
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Lemaitre, G., Nogueira, F., & Aridas, C. K. (2016). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning.
- Louppe, G., Wehenkel, L., Suter, A., & Geurts, P. (2013). Understanding variable importances in forests of randomized trees. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 26). Curran Associates, Inc. [https://proceedings.neurips.cc/paper\\_files/paper/2013/file/e3796ae838835da0b6f6ea37bcf8bcb7-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2013/file/e3796ae838835da0b6f6ea37bcf8bcb7-Paper.pdf)
- Lundberg, S., & Lee, S.-I. (2017). A unified approach to interpreting model predictions.
- Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S.-I. (2019). Explainable ai for trees: From local explanations to global understanding.
- Luzzatto, A., & Haridim, M. (2016). *Wireless transceiver design: Mastering the design of modern wireless equipment and systems*. John Wiley & Sons.
- Maharana, K., Mondal, S., & Nemade, B. (2022). A review: Data pre-processing and data augmentation techniques [International Conference on Intelligent Engineering Approach(ICIEA-2022)]. *Global Transitions Proceedings*, 3(1), 91–99. <https://doi.org/10.1016/j.gltp.2022.04.020>
- Marcinkevičs, R., & Vogt, J. E. (2023). Interpretable and explainable machine learning: A methods-centric overview with concrete examples. *WIRES Data Mining and Knowledge Discovery*, 13(3), e1493. <https://doi.org/10.1002/widm.1493>
- Marques de Sá, J. P. (2003). Statistical classification. In *Applied statistics using spss, statistica and matlab* (pp. 191–235). Springer Berlin Heidelberg. [https://doi.org/10.1007/978-3-662-05804-6\\_6](https://doi.org/10.1007/978-3-662-05804-6_6)
- Mdini, M., Simon, G., Blanc, A., & Lecoeuvre, J. (2020). Introducing an unsupervised automated solution for root cause diagnosis in mobile networks. *IEEE Transactions on Network and Service Management*, 17(1), 547–561. <https://doi.org/10.1109/TNSM.2019.2954340>
- Mohammed, M., Khan, M. B., & Bashier, E. B. M. (2016). *Machine learning: Algorithms and applications*. Crc Press.
- Mohanani, R., Salman, I., Turhan, B., Rodríguez, P., & Ralph, P. (2020). Cognitive biases in software engineering: A systematic mapping

- study. *IEEE Transactions on Software Engineering*, 46(12), 1318–1339. <https://doi.org/10.1109/TSE.2018.2877759>
- Molnar, C. (2022). *Interpretable machine learning: A guide for making black box models explainable* (2nd ed.). <https://christophm.github.io/interpretable-ml-book>
- Neyman, J. (1992). On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. In S. Kotz & N. L. Johnson (Eds.), *Breakthroughs in statistics: Methodology and distribution* (pp. 123–150). Springer New York. [https://doi.org/10.1007/978-1-4612-4380-9\\_12](https://doi.org/10.1007/978-1-4612-4380-9_12)
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Pintelas, E., Livieris, I. E., & Pintelas, P. (2020). A grey-box ensemble model exploiting black-box accuracy and white-box intrinsic interpretability. *Algorithms*, 13(1). <https://www.mdpi.com/1999-4893/13/1/17>
- Potdar, K., Pardawala, T., & Pai, C. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, 175, 7–9. <https://doi.org/10.5120/ijca2017915495>
- Powers, D. (2008). Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation. *Mach. Learn. Technol.*, 2.
- Qlik. (2024). Data wrangling [Accessed: 2024-05-07]. <https://www.qlik.com/us/data-management/data-wrangling>
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938. <http://arxiv.org/abs/1602.04938>
- Robnik-Šikonja, M., & Kononenko, I. (2008). Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5), 589–600. <https://doi.org/10.1109/TKDE.2007.190734>
- Roelofs, R., Shankar, V., Recht, B., Fridovich-Keil, S., Hardt, M., Miller, J., & Schmidt, L. (2019). A meta-analysis of overfitting in machine learning (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett, Eds.). [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/ee39e503b6bedf0c98c388b7e8589aca-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/ee39e503b6bedf0c98c388b7e8589aca-Paper.pdf)
- Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.

- Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), 1–21. <https://doi.org/10.1371/journal.pone.0118432>
- Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*. <https://doi.org/10.1007/s42979-021-00592-x>
- Seo, S. (2006). A review and comparison of methods for detecting outliers in univariate data sets.
- Siddiqi, M. A., Yu, H., & Joung, J. (2019). 5g ultra-reliable low-latency communication implementation challenges and operational issues with iot devices. *Electronics*, 8(9). <https://doi.org/10.3390/electronics8090981>
- SSB. (2023). Befolkning på rutenett 1000 m 2023. <https://kartkatalog.geonorge.no/metadata/befolkning-paa-rutenett-1000-m-2023/8de78b6a-6634-40f2-aac1-954d82ec31b7>
- Szilagyi, P., & Novaczki, S. (2012). An automatic detection and diagnosis framework for mobile communication systems. *IEEE Transactions on Network and Service Management*, 9(2), 184–197. <https://doi.org/10.1109/TNSM.2012.031912.110155>
- Tanimoto, A., Yamada, S., Takenouchi, T., Sugiyama, M., & Kashima, H. (2022). Improving imbalanced classification using near-miss instances. *Expert Systems with Applications*, 201, 117130. <https://doi.org/https://doi.org/10.1016/j.eswa.2022.117130>
- Telenor. (2021). Store gevinster for industrien og samfunnet.
- Turner, D., Levchenko, K., Mogul, J., Savage, S., & Snoeren, A. (2012). On failure in managed enterprise networks. *HP Laboratories Technical Report*.
- Uddin, M. S., Ahmmad Bhuiyan, E., Sarker, S., & Das, S. (2021). An intelligent short-circuit fault classification scheme for power transmission line. <https://doi.org/10.1109/ACMI53878.2021.9528200>
- Van den Berg, J., Litjens, R., Eisenblätter, A., Amirijoo, M., Linnell, O., Blondia, C., Kürner, T., Scully, N., Oszmianski, J., & Schmelz, L. C. (2008). Self-organisation in future mobile communication networks. *Proceedings of ICT-Mobile Summit 2008, Stockholm, Sweden, 2008*.
- Van Hulse, J., Khoshgoftaar, T. M., & Napolitano, A. (2009). An empirical comparison of repetitive undersampling techniques. *2009 IEEE International Conference on Information Reuse Integration*, 29–34. <https://doi.org/10.1109/IRI.2009.5211614>
- Webb, G., & Ting, K. (2005). On the application of roc analysis to predict classification performance under varying class distributions. *Ma-*

- chine Learning*, 58, 25–32. <https://doi.org/10.1007/s10994-005-4257-7>
- Węglarczyk, Stanisław. (2018). Kernel density estimation and its application. *ITM Web Conf.*, 23, 00037. <https://doi.org/10.1051/itmconf/20182300037>
- Wei, W., Li, J., Cao, L., Ou, Y., & Chen, J. (2013). Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4), 449–475. <https://doi.org/10.1007/s11280-012-0178-0>
- What is regularization? (n.d.). *IBM*. <https://www.ibm.com/topics/regularization>
- Whitfield, B. (2023). The importance of feature engineering in machine learning [Accessed: 2024-05-07]. <https://builtin.com/articles/feature-engineering>
- Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering* (1st ed.). Springer Berlin, Heidelberg. <https://doi.org/10.1007/978-3-662-43839-8>
- Wrålsen, A., & Berntsen, K. E. (n.d.). *Vitenskapelighet i ba-oppgaven* [This article is part of the syllabus of writing the bachelor thesis].
- Zhan, C., Zheng, Y., Zhang, H., & Wen, Q. (2021). Random-forest-bagging broad learning system with applications for covid-19 pandemic. *IEEE Internet of Things Journal*, 8(21), 15906–15918. <https://doi.org/10.1109/JIOT.2021.3066575>



## **A. Interview with Experts**

# **Bachelor**

## Semi-Structured Interview with Telenor

10:15-11:00 April 17, 2024

### Attendance

Interview Guide: Trym Hamer Gudvangen

Present: Trym Hamer Gudvangen, Callum Gran, Nicolai Hollup Brand, Azza H. Ahmed, Pablo Ortiz, David Zsolt Biro, Eirik Hoel Høiseth.

In this interview, the interviewee names will be highlighted as following for ease of readability:

- Trym
- Pablo
- David
- Eirik
- Azza

Meeting starts 2 minutes late, all are present.

In this interview, we have decided to use informal pronouns to ensure that no one is directly misquoted. This comes from the fact that we lack the ability to write fast enough to ensure 100% correctness of each statements. An effect of this, is that some paraphrasing has been made, but keeping the correctness was of utmost importance.

### **Introduction of Group:**

**Trym** introduces the bachelor group: Hi, We are three bachelor students studying computer science at NTNU. For our thesis, we are collaborating with SimulaMet on researching fault detection systems for the 3GPP self-organizing network specification. Specifically, we are investigating fault detection from the end-user's point of view rather than intra-network. Through real data collected by NorNet Edge nodes, we are developing models with a focus on acceptable performance, low overhead, and high explainability.

**Before we start, are you comfortable with having your name in our bachelor thesis or would you like to be anonymous?**

All: No worries. It would be nice for them to review the included interview parts prior to publishing. We should also send the thesis when we are done with the bachelor.

**Could you please tell us a little about your background and role in Telenor?**

**Pablo:** PhD in Theoretical Physics. He has been working at Telenor since 2018, and since 2020 in the research department. His work is mostly within speech and language technologies, but he has later been working on time series data for networks the past 2-3 years.

**David:** Masters in Electrical Engineering. He has been working at Telenor in Hungary since 2012. Previously, he worked with radio network as a solution network architect for 4G in Denmark. 5-6 years ago his role changed to data analytics, specifically within big data and machine learning to figure out patterns in the network data. Currently, he works with anomaly detection and “closed-loop” monitoring.

**Eirik:** PhD in Mathematics at NTNU. Previously he worked as an AI consultant, but has worked at Telenor since 2021. His work has mostly been within intelligent power saving. Although, the last year and a half he has been working on anomaly detection for time series data within different parts of Telenor.

**Azza:** A postdoc at SimulaMet with a PhD from Oslo-Met. She works on improving mobile networks and similar radio technologies. Currently she has been working on anomaly detection in time series and usage of different AI models within networks. She is also the supervisor for this bachelor thesis.

### **How familiar are you with 3GPP's Self-healing specifications?**

**Pablo:** Not well known. More works with AI side.

**David:** Used to be somewhat familiar.

**Eirik:** Not well known. More works with the AI side.

### **What systems are currently utilized today to detect and diagnose faults in the mobile broadband network? How do they work?**

**David:** Telenor Denmark outsources the radio operations to an external party that has a system from Nokia with SON capabilities. Virtual drive testing, it is based on timeseries that are collected and visualized, but it has an expert system based on human based rules. Right now, they are trying to implement more and more machine learning models to relieve the experts. They have a data lake, and then they use python to infer data and they process this on to the alarming system that delivers alarms to their platforms. They use NPS scores, where they send out SMS to customers to get feedback about the subjective experience, which is basically surveys about how they feel the data is. They are trying to move from the network point of view to the customer point of view. There is some probing data and the postcodes of the customers, but they aren't looking at the customers specifically, they are trying to aggregate up so they can look at customers from a larger point of view. In short, trying to look at a higher level of aggregated data to give the best average experience. However, they are now trying to look into tools to try and see more customer specific data, but this isn't quite there yet and is only in the idea phase. In Denmark the radio network is a rule based system for fault detection, but the core network uses a Gaussian system.

**Eirik:** It would be mostly speculation to talk about, as the department that Pablo and Eirik work for doesn't really work directly with the network.

### **To which extent do the current mobile networks today implement this specification?**

Question is skipped as it wasn't deemed relevant based on the previous answers.

### **What are some of the problems with today's solutions for fault detection and diagnosis?**

**David:** False positive rates are a problem with the systems. Important to differ between false positives within mathematical and business points of view. If you have one single spike, this is an error in math, but not necessarily a fault in the business. If you react too early, you may cause other spikes. Sometimes it makes sense to look into spikes and see if they repeat at the same times or under the same circumstances. The most important thing is to understand if it is worth it for the business to actually act on the anomalies that are detected. Marking spikes is the first step, if these spikes need to be acted on is the next.

**Pablo:** See the spike a posteriori. If a spike is something you have recovered from, you have to see if this causes any other issues.

**Eirik:** Simple thresholds miss the more complex situations.

### **How does Telenor collect feedback and data surrounding faults and experiences of the end-user? How do you quantify the user experience?**

**David:** Mentioned surveys where subjective feedback is given. Facebook analytics. Ookla has some data that they give. There are also some probing systems which give signaling data. In Europe, they mainly collect larger data, as they cannot collect individual data. The user plane is not allowed to be collected. In Asia, it is more open to collect data from the user plane. Data is mostly connected from probes that are in the network. Yes, there are some tools, but in reality everything is aggregated. If you see faults on the radio network, this may be because of switching between the IP network etc. If there is a radio fault, then it will be a failure in the core network, and there are a lot of people pointing at each other. It is important to keep a holistic view, as this is a network.

**Pablo:** Not in production, but in the pipeline they want to use data from customer service calls. Not data from the calls, but data about the amount of calls and the categories of the calls.

**Eirik:** Telenor values the Ookla speedtest and being on top there, but this is not really a good measure of user experience.

**Trym** and **Azza:** Explains what NordNet Edge is and the research.

**David:** We have data that is the other way around of what he is working with. They collected data from probes in taxi cabs. In Denmark they are implementing thousands of FWA nodes.

### **To what degree of autonomy, can fault detection and diagnosis systems be used in Telenor? In other words, if a fully autonomous system existed, would it be implemented?**

**David:** "Impact number", how many customers you are affecting. Fully autonomous to the core network from the radio network we are far from it. From the radio network it is possible

to go fully autonomous. Features and thresholds are optimized and running autonomously (despite not being ML). Less feedback loop, but more open-loop ideas. Go into the direction of autonomy. At the radio side, you are not affecting that many customers. Impact number is an important measure. Generally, it is very important to consider the human aspects before implementing an autonomous solution. It needs human supervision. It also needs a “kill” switch. Model accuracy is not so important, what is important is good average customer satisfaction.

**Pablo:** In principle, fault detection and diagnosis is just providing information. A long distance from making this information able to be used autonomously to make adjustments. There are different levels of severity from the business and customer point of view. An autonomous system must put numbers on things, so considering fully autonomous systems is very complicated.

**Eirik:** “Always” is a hard word. But for now I don’t think they are willing to go fully autonomous. They need a human in the loop. The network is defined as critical architecture so it may not even be legal to go 100% AI. Telenor Norway is interested in using AI for making steps to go MORE autonomous, but fully is many years in the future.

**Follow-up question: How important is the explainability of a potential self-healing solution? Trym describes categories of models into classes based on explainability.**

**David:** Can you make problem resolution without explaining what caused the problem? You need some explainability, but not 100%. One value in, one value is not going to be trusted. Such models are not really useful.

**Pablo:** Doesn’t understand the other class, but understands the class of ML models that you can use post-hoc methods (shap values, MDI). In addition they use some attention based networks, where you can get some information on where they put their attention. It is super important to be able to understand how a fault is detected, so they can be deciphered. There is a sweet spot between a good enough model so that you can trust the results, but that is also simple enough to explain. People working with data and algorithms understand there are errors, but business people don’t necessarily take this into consideration. Meaning they could read model values as hard facts.

**Eirik:** It is very important! Ericsson has a whole team working on explainability within AI. Ideally people would like to use simple models, but the complexity makes it necessary to use more complex models. Use simple models as a baseline. You need to prove you need the complexity.

**Would an automatic solution based on end-user data, as described at the start of the interview, be interesting to Telenor? Do you see any potential value in this? If so, to what extent etc...**

**David:** Yes, of course it could be relevant. Actually, I think it's more relevant from a design point of view. How capacity is in certain places. Including end-user measurements could be used to verify intra-network measurements. David asks our thoughts on what our ML solution looks like.

**Trym:** Answers by describing we are detecting faults based on RTT while trying to reduce overhead as much as possible.

**David:** His understanding is that we run the model on the equipment. David's approach would not run on real-time data, but rather on a lot of historical data. David is skeptical towards our approach. Too short of a time frame to be able to react from the network side. This data is already available on the E-nodeB side. All nodes for an area need to know each other so they work in accordance with each other. As if one station changes its coverage, that coverage will need to be overtaken by another.

**Pablo:** That kind of data, it's pretty useful. Some of this data already exists from crowdsourcing. Agrees with David that collecting data long term is interesting, but short term maximizing networks may not. There are some pilot projects on self optimization for electricity saving. It may be idealistic to keep updating the network constantly, and network operators may be hesitant. Pure network people generally agree with David.

**Do you have any additional comments/questions/suggestions?**

All: No

