TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Kaarel Koovit 221953IVEM

Machine Learning Based Modelling of TCP Data on Raspberry Pi

Master's thesis

Supervisors: Yannick Le Moullec
PhD
Kanwal Ashraf
MSc

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kaarel Koovit 221953IVEM

Masinõppel põhinev TCP andmete modelleerimine Raspberry Pi-l

Magistritöö

Juhendaja: Yannick Le Moullec
PhD
Kanwal Ashraf
MSc

Tallinn 2023

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kaarel Koovit

02.01.2024

Abstract

In telecommunication systems, the requirements for network optimization and scheduling quality are constantly increasing. One solution is to use data-driven machine learning (ML) instead of a model-based approach to control the optimization and scheduling of a network. In this thesis, the suitability of machine learning is explored as a first step towards developing a network optimization algorithm that is suitable to run on resource constrained devices (e.g., Raspberry Pi), that will use information obtained when monitoring the activities of the transport layer in the OSI network architecture model.

The suitability of both unsupervised and supervised learning methods is investigated, i.e. PCA and k-means for unsupervised learning, and linear regression for supervised learning. For the analysis, a dataset is extracted from a network setup consisting of up to five Raspberry Pi (RPi) Pico W-s connected to a RPi 4 over Wi-Fi, using *Wireshark* to capture the transmission control protocol (TCP) trace data.

It is found that unsupervised learning is limited in separating clusters or finding principal components that would categorize the network setup. On the other hand, it is found that the linear regression model predicts a determination coefficient of 66% when using three parameters for the regression analysis and the combined dataset of different induced network constraints, and a 99,75% determination coefficient for a non-constrained network dataset; these coefficients are used to predict the number of devices connected to the network. The minimal amount of data required to develop a (near) optimal algorithm is found to be dependent on the dataset used for the algorithm training.

The final algorithm is deployed onto the RPi to monitor the network traffic and predict the number of devices connected to the network. Experimental results show an overall accuracy of 94,67% for an unconstrained network and a capture time window of 10 s, which is close to the model's prediction.

This thesis is written in English and is 89 pages long, including 6 chapters, 44 figures and 16 tables.

Annotatsioon

Masinõppel põhinev TCP andmete modelleerimine Raspberry Pi-l

Nõuded võrgu optimeerimise ja planeerimise kvaliteedi jaoks telekommunikatsioonisüsteemides kasvavad pidevalt. Üks lahendus on kasutada andmepõhist masinõpet mudelipõhise lähenemise asemel, juhtimaks võrgu optimeerimist ja planeerimist. Selles töös vaadeldakse masinõppe sobivust esimese osana töötamaks välja võrgu optimeerimise algoritmi, mis töötaks piiratud ressurssidega seadmetel (näiteks Raspberry Pi). Algoritmi väljatöötamiseks kasutatakse OSI võrguarhitektuurist transpordikihi andmeid.

Algoritmi väljatöötamiseks uuritakse nii juhendamata kui ka juhendatud masinõppe meetodeid – PCA ja k-means algoritmid juhendamata ja lineaarne regressioon juhendatud masinõppe jaoks. Analüüsi jaoks kogutakse andmed jälgides liiklust võrgus, mis koosneb kuni viiest Raspberry Pi Pico W-st ja Raspberry Pi 4-st, mis on omavahel Wi-Fi'ga ühendatud. TCP andmete jälgimiseks kasutatakse programmi *Wireshark*.

Juhendamata masinõppimise puhul leitakse, et on piiratud edu klastrite eraldamises või peamiste komponentide leidmises, mis kirjeldaks võrgu ülesehitust. Teisest küljest leitakse, et lineaarse regressiooni modelleerimine ennustab määramiskoefitsiendiks 66%, kui kasutatud on kolm parameetrit analüüsi jaoks ja kombineeritud andmekogum sisaldab erinevaid võrku sisse viidud piiranguid, ja 99,75% kui kasutatud on piiramata võrgu andmekogum, ennustamaks võrku ühendatud seadmete arvu, kasutades leitud koefitsiente. Minimaalne andmete kogum, mis on vajalik optimaalse algoritmi väljatöötamiseks, sõltub andmetest, mida on mudeli häälestamiseks kasutatud.

Lõplik algoritm käivitatakse RPi 4 peal, jälgimaks võrgu liiklust ja ennustamaks võrku ühendatud seadmete arvu. Eksperimentide tulemus näitab algoritmi 94,67% täpsust piiramata võrgu ja 10 s ajavahemiku jaoks, mis on lähedal väljatöötatud mudeli täpsusele.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 89 leheküljel, 6 peatükki, 44 joonist, 16 tabelit.

Acknowledgment

I express my gratitude to both my supervisors and also to Senior Researcher Andrei Krivošei for his support regarding the data analysis, in particular the linear regression method. Our discussions provided additional insights on how to approach the topic.

List of abbreviations and terms

AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Access Point
DL	Deep Learning
GNN	Graph Neural Network
KPI	Key Performance Indicator
LR	Linear Regression
MAE	Mean Average Error
ML	Machine Learning
PCA	Principal Component Analysis
PDF	Probability Density Function
RF	Random Forest
RMSE	Root Mean Square Error
RPi	Raspberry Pi
RTT	Round Trip Time
SBC	Single Board Computer
SGA	Stochastic Gradient Ascent
SGD	Stochastic Gradient Descent
SON	Self-Organizing Network
TCP	Transmission Control Protocol
URLLC	Ultra Reliable Low Latency Communication

Table of contents

List of figures

List of tables

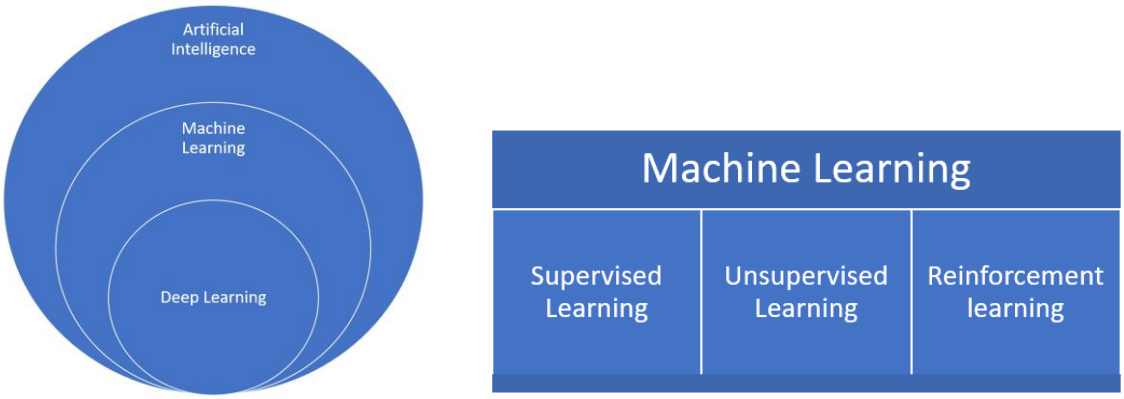
1 Introduction

In telecommunications, the requirements to the networks' parameters are increasing rapidly with the introduction and implementation of more complex, higher throughput technologies [1]. The classical model-based approach used to optimize and control the network parameters are proving to be increasingly more difficult to apply due to the complexity and heterogeneity of the networks, and the difficulty to obtain the system parameters and lossless block decomposition [2]. Therefore, Machine Learning (ML) based data-driven methods, i.e. data-driven machine learning (DDML), are increasingly preferred to exploit the online and offline data for the controller design and network optimization.

Network optimization involves monitoring and improving the key process indicators (KPIs) of a network. The parameters that are optimized can include bandwidth (data transfer rate), delay (the time a packet travels from source to destination), loss (the amount of data lost), jitter (variation in data transfer rate), transmission power (power required to transmit data with acceptable loss). The goal of network optimization is to find a balance between these parameters based on the amount of traffic in the network and e.g. target quality of service. To be able to optimize a network, the network elements and their relationships need to be defined and this can be done either by using a model-based approach (creating the model first based on artificial assumptions) or data-driven approach (building the model using available data).

Indeed, the classical model-based approach is proving more difficult to apply due to the increase in network complexity and even though a network is divided into multiple layers, attaining optimal or near-optimal results may prove difficult due to dependencies, including opposing ones (for example channel modulation, detection and coding, which should be separately optimized) among those; such compromising problems are further amplified in 5G and beyond.. By using DDML methods, it is possible to use the data from the whole network instead of the relation of the network elements to predict different patterns and scenarios. Figure 1 (a) shows an overview of ML in relation to artificial

intelligence (AI). AI involves the concept of using computer systems to perform tasks that require human intelligence. ML is a part of AI and it involves using different techniques to enable the computers to learn or improve on a task based on experience without relying on explicit instructions. A subset of ML is deep learning (DL) which uses algorithms that have an artificial neural network (ANN) structure that consists of multiple layers. In ML, there exist many methods which can be divided into supervised, unsupervised, and reinforcement learning. Figure 1 (b) illustrates the three different types of ML, with a more detailed overview provided in Chapter 3. For network optimization and scheduling improvement, methods from all three different areas have been used and reported in the scientific literature.



(a) (b)

Figure 1. (a) relation between AI, ML and DL, (b) the three machine learning types.

The main developments so far have been to exploit the quantity of available data to develop the algorithms [3]; however, not much research has been carried out for determining the minimum amount of data required to develop the optimal operation algorithms [4]. Additionally, most of the research has focused on the development of the optimization algorithms but has not delved much into implementing the algorithms, for example on different resource constrained devices [5].

1.1 Research Statement

To control and possibly optimize the network performance by reducing losses in the network, network congestions, power consumption, and by improving the bitrate, a light-

weight algorithm utilizing the data-driven technique should be identified and optimized. This thesis focuses exclusively on the transport layer and the layers above it in the OSI network architecture model when acquiring and analysing the data and developing the network optimization techniques. The algorithm should be suitable to run on devices with low computing resources and power consumption, such as single board computers; one potential application for the developed algorithm could be a portable bioanalytical device (for example the PRG620 project-based flow cytometry device). The quality and quantity of the data (minimum amount of data) required for the algorithm to achieve the desired performance should also be evaluated.

Given the above, the three research questions of the thesis are expressed as:

- **RQ1: As a first step towards network performance control and (near) optimal scheduling, can there be proposed an algorithm for predicting the occurring network constraints or the number of client devices towards using unsupervised (e.g., PCA) and supervised learning (e.g., linear regression) methods?**
- **RQ2: Can the minimal amount of data required for (near) optimal scheduling be identified?**
- **RQ3: Can the algorithm be deployed on an SBC (for example Raspberry Pi) and what will be its impact on the performance?**

1.2 Thesis organization

Chapter 1 provides an overview of the motivation of the work and the research questions.

Chapter 2 gives a state-of-the-art overview of the DDML-based techniques for network optimization and control.

Chapter 3 contains an overview of the background of data-driven techniques, network optimization, and how data-driven techniques are used for network optimization and transmission control protocol.

In Chapter 4, the design and implementation of the proposed DDML algorithm, the selection of the dataset and the hardware platform are explored.

The corresponding experimental results and the analysis thereof are given in Chapter 5.

Finally, Chapter 6 provides a conclusion for the performed work, lessons learned, and suggestions for future research directions.

2 State of the Art Overview

This chapter provides an overview of the recent trends in data-driven techniques that can be used for network control and optimization, and for using TCP data for network parameter analysis.

During the past few decades, the rapid development of communication technologies has been accompanied by a need for improved techniques for network control, scheduling and optimization [3]. One of the proposed directions for the development of network control techniques is to use DDML. Several works have been published which discuss possible approaches for data-driven network optimization, control, and scheduling using different machine learning methods and acquired datasets. This chapter gives an overview of representative examples of such works and highlights their overall main limitations, which this MSc thesis aims to address.

In [5], Salh et al. provides an overview of different ML techniques that can be used for network optimization – see Table 1. The main technique to be improved that is discussed for the application of ML-based solutions is Ultra-Reliable and Low Latency Communication (URLLC). For example, support vector machine (Table 1 – row 5) from the supervised learning technique was proposed to improve the prediction of propagation path loss in wireless networks; the self-organizing map (Table 1 – row 11) from the unsupervised learning technique was proposed to improve the capacity and user experience in small cells; and Q-learning (Table 1 – row 14) from the reinforcement technique was proposed to provide the users with a better ability to predict the return function in order to improve data rate. The paper explores URLLC and its enhancements and suggests further research directions, i.e. computation efficiency, hardware development for 6G, scalability and robustness, THz communication, energy management, channel estimation. While the above work provides suitable options from different ML model types out of the proposed ML techniques (supervised, unsupervised and reinforcement learning), this thesis mainly focuses on unsupervised and supervised learning to investigate the possibility of using these techniques for solving the proposed problem, in order to determine the possibility of using these techniques on an SBC and

because the supervised learning (regression models) and unsupervised learning (dimensionality and clustering models) provide a useful comparison on the performance of different approaches. Furthermore, unsupervised learning is featured less frequently in papers [6], which provides an opportunity to explore potential new approaches.

Table 1. Summary of ML techniques and their applications in wireless communication, adapted and expanded from [5].

ML Technique	Learning Model	Mobile and Wireless Communication	Year
Supervised Learning	Self-Organizing Networks (SON)	Optimizes network management location, maximizes the capacity [7]	2017
	Linear reversion	Facilitates energy harvesting and prediction by equipping the harvesting node with adaptation to the current energy using real-time power measurements [8]	2016
	Supervised Classifier	Enables autonomous network management aware of quality of experience to improve prediction of the network demand and network sensitivity [9]	2018
	Support Vector Machines	Predicts propagation path loss in wireless networks [10]	2015
	Quantum ML	Increases performance through enabling technologies at the network edge, air interface, and on the user's end [11]	2019
Unsupervised Learning	Latent function with unsupervised DL	Reduces problems regarding the QoS constraint of URLLC [12]	2019
	K-means clustering	Neural-network prediction to provide increased capacity for users [13]	2017
	K-means technology	Low latency data access and storage of data blocks using DL [14]	2019
	Unsupervised clustering	Used to decide low and high-power node allocation to reduce latency and power [15]	2018
	Self-organizing map	Increases capacity and improves user experience for coverage planning and performance optimization [16]	2018
Reinforcement learning	Framework dynamically predicts	Guarantees long term reliability and latency for every user, by balancing E2E reliability, latency, and data rate [17]	2019

ML Technique	Learning Model	Mobile and Wireless Communication	Year
Reinforcement learning	Markov decision process	Enables vertical handoff decisions based on network parameters, e.g. minimum bandwidth, delay, and battery level of terminal [18]	2008
	Q-learning	Maximizes data rate and enables users to predict their return function [19]	2018
	Deep Q-network	Increases Signal-to-Noise-Plus-Interference Ratio (SNIR) and efficacy, selects optimal anti-jamming communication policy [20]	2017
	Deep-RL	Reduces system cost by applying joint optimum caching and estimating allocation [21]	2018
	Dueling deep-Q	Improves probability of QoS level approval, data rate of the network and learning effectiveness [22]	2022
	Deep deterministic policy gradient (DDPG)	Reduces energy costs while still ensuring that there are no unacceptable delays [23]	2022

Furthermore, in [24], Ma et al. gives an overview of data-driven 5G network optimization techniques with ML, including proposals for different network traffic prediction models and correlation to network KPIs. Table 2 shows the different proposed ML models for network prediction.

Table 2. Network traffic prediction ML models, adapted and expanded from [24].

Model	Traffic type	Prediction	Type	Reference	Year
Deep learning	Cellular	Temporal	Network level	[25]	2017
LSTM	Cellular	Temporal		[26]	2018
Exponential smoothing	Cellular	Temporal + spatial (TS)	Cell level	[27]	2007
Statistics	Cellular	TS		[28]	2011
ARMA	Cellular	TS		[29]	2016
α-stable	Online	TS		[30]	2017
LSTM	Cellular	TS		[31]	2018
Neural network, GP	Cellular	TS		[32]	2018
Markov chain	Cellular	TS	Online data	[33]	2011
ARMA, decision tree	Cellular	TS		[34]	2017
Statistics	Online	TS		[35]	2016
RF-GRU-NTP	Online	TS		[36]	2022
Regression	Online	TS		[37]	2015
Random forest	Online	Temporal		[38]	2021
k-means, neural network	Online	TS	Anomaly detection	[39]	2017
k-means, GP	Cellular	TS		[40]	2018
k-means, NARX	Online	TS		[41]	2018

The work presented in [2] provides an overview of why data-driven networks are advantageous over model-based solutions. Data-driven networks do not require exact network models, have a uniform architecture which makes them easier to transpose, are less sensitive to system parameters and do not require block decomposition (dividing the network into multiple layers, which in turn are consisting of separate blocks). In the paper, a typical use scenario is provided for data-driven control, optimizing network load balancing using machine learning that uses a sequence of successive predictions. In [4],

Baggio et al. proposes a data-driven network control method, using finite data, to optimally direct the network nodes' state to a desired one within a finite-time space. Their experiments show that the data-driven control has advantages compared to model-based solutions, i.e. the results show that the computational time can be approximately 10 to a 100 times smaller for the data-driven control compared to the model-based one, depending on the network size and the final stage error for a fixed amount of control nodes, and increasing network size can be up to 10^5 times smaller for the data-driven control method, and it can be applied for controlling actual networks. The limitations and the possibility for future improvements of the provided solutions are that the results are provided with the assumption that the dynamics of the networks are linear, the results were calculated with engineering and scientific routines (for which the precision could be improved upon), a point-to-point control strategy was used instead of a closed-loop one, and the reconstruction error is not provided with a non-asymptotic guarantee. In [42], Zhang et al. gives potential challenges for using DDML for network management (in the form of SON) – data imbalance, data insufficiency, cost insensitivity, non-real-time response, multisource data fusion. Additionally, they suggest potential solutions and related methods, including unsupervised learning. Furthermore, a case study is provided, illustrating the relevance of the proposed methods.

These above works demonstrate that the use of DDML methods for complex networks where the network dynamics are unknown is advantageous over a model-based approach, but some potential disadvantages need to be taken into consideration. In what follows, works related to ML techniques used in network control and optimization are presented.

In [12], Sun et al. proposes a framework, using unsupervised deep learning, to learn the underlying function of a system, which is applicable in both variable and functional optimization problems and demonstrates this solution for a URLLC optimization problem. The DNN is changed to unsupervised learning by using the property itself (e.g. optimal bandwidth allocation) instead of the label of the property and the proposed optimization methods are stochastic gradient descent (SGD) and stochastic gradient ascent (SGA). Using training parameters of 1000 times training, with 10000 iterations each, and a training reliability requirement of $\varepsilon_D = 10^{-6}$ (with the overall reliability being $\varepsilon_{max} = 10^{-5}$), then the availability achieved was 98,9% and bandwidth loss was 3,3%. Additionally, in [43], Sun et al. proposes an unsupervised deep learning approach also based on SGD for solving a QoS constrained optimization problem for bandwidth and

power. Their results show that the ML-based solution can achieve the same performance as the optimal solution and outperforms existing policies, by saving 40% of the bandwidth consumption. In [44], Ferriol-Galmés et al. proposes graph neural network (GNN) model for network optimization that can estimate QoS parameters. The custom GNN model, named TwinNet, consists of message-passing and readout modules and is able to achieve a mean absolute percentage error (MAPE) of 6,3% with a real test application. In [45], Farthofer et al. provides a dataset of mobile drive tests, a tool chain for the data analysis and illustration for its usage. The suggested analysis method involves a feed forward neural network and autoencoder neural network to analyse the data and different scenarios are simulated to demonstrate the behaviour and results of the network. In [1], Jaffry et al. suggests an approach for network anomaly detection, using a data-driven method, which involves iteratively comparing the probability density function (PDF) to a threshold value and then recalculating the PDF. Their algorithm displayed a high accuracy of 98,08% between neighbouring grids and reduced accuracy the further away the predicted grid was. In [46], Delimargas et al. explores the possibility of using principal component analysis (PCA) for network trace data analysis and anomaly detection. As the classical PCA is prone to giving false positive and false negative results because of the method's sensitivity to noise, then the PCA algorithm has been modified to overcome this problem. Their results show that the algorithm is in some cases able to detect TCP network scan anomalies with an intensity (defined in the paper as the ratio of flows with anomalies to the average number of flows in a time window) of 5% and in most cases with an intensity of 20%.

For the TCP data analysis, in [47], Chaudry explores the possibility of using ML to find hidden connections between round-trip time (RTT) and the throughput for Wi-Fi connections. The PCA, linear regression and random forest (RF) techniques are used to explore these relationships. For their analysis, the physical and data-link layer parameters' measurements were extracted from the available dataset. Their results show that RTT alone might not be enough to predict the Wi-Fi throughput, but there is indirect confirmation that RTT can have a significant impact on Wi-Fi throughput. The PCA shows that RTT has a variance ratio of 0,876 for the principal component (PC) 1, that itself had a proportion of variance of 0,984, when used with the RF method. In [48], Arlitt et al. research the possibility to predict latency from information obtained from TCP traces for short transfers (defined by transfer length in bytes). Their results show that,

depending on the maximum segment size (MSS), the RTT has a correlation to bandwidth of up to -0,370 and with latency up to 0,511, depending on the maximum segment size (MSS) and the trace file used. This suggests that RTT alone might not be enough to predict bandwidth or latency and other parameters from the traces could be included for the analysis.

Despite their valuable contributions, the solutions and proposed methods provided in the above papers do not consider the application of the optimization and scheduling algorithms on a low power device (e.g., RPi), which is why additional research is required in this direction; this is one of the goals of this thesis. The base chosen to be explored for developing the algorithm to be used for network control on a low-powered device were unsupervised learning, as it gives the possibility to use unlabelled network trace data for optimizing the network parameters, and linear regression.

The main algorithms chosen for the network data analysis from unsupervised learning were principal component analysis (PCA) and k-means because these algorithms provide dimensionality reduction for extensive trace data, reduce the computational time needed for the analysis and will be able to provide a basis for the final data analysis, to indicate the final algorithms suitable for characterize the data.

This chapter gave an overview of recent trends in the DDML techniques that can be used for network optimization and control and the TCP data analysis possibilities. The next chapter presents the details of the background theory for DDML techniques, network optimization, using DDML techniques for network optimization and TCP.

3 Background Theory Overview

3.1 Data-driven techniques

AI is the intelligence of machines or software, a field of study that develops and studies intelligent machines. ML is a part of AI and studies the programs that can improve the performance of a task automatically, by discovering their own algorithms.

ML is divided into three main approaches, based on the learning paradigms: supervised learning, unsupervised learning, and reinforcement learning.

In brief:

Supervised learning uses labelled data, with given inputs and corresponding outputs to learn the rule of how the input and output are related.

Unsupervised learning does not have labels on the data and the algorithm should develop the relation of the data itself, by finding patterns in the data or learning the features behind the data.

Reinforcement learning involves the algorithm that interacts with an environment dynamically in order to learn to achieve a certain goal and based on feedback from the actions performed, the program tries to maximize the rewards.

An overview of selected possible DDML techniques by learning type is provided in Figure 2, adapted from [49].

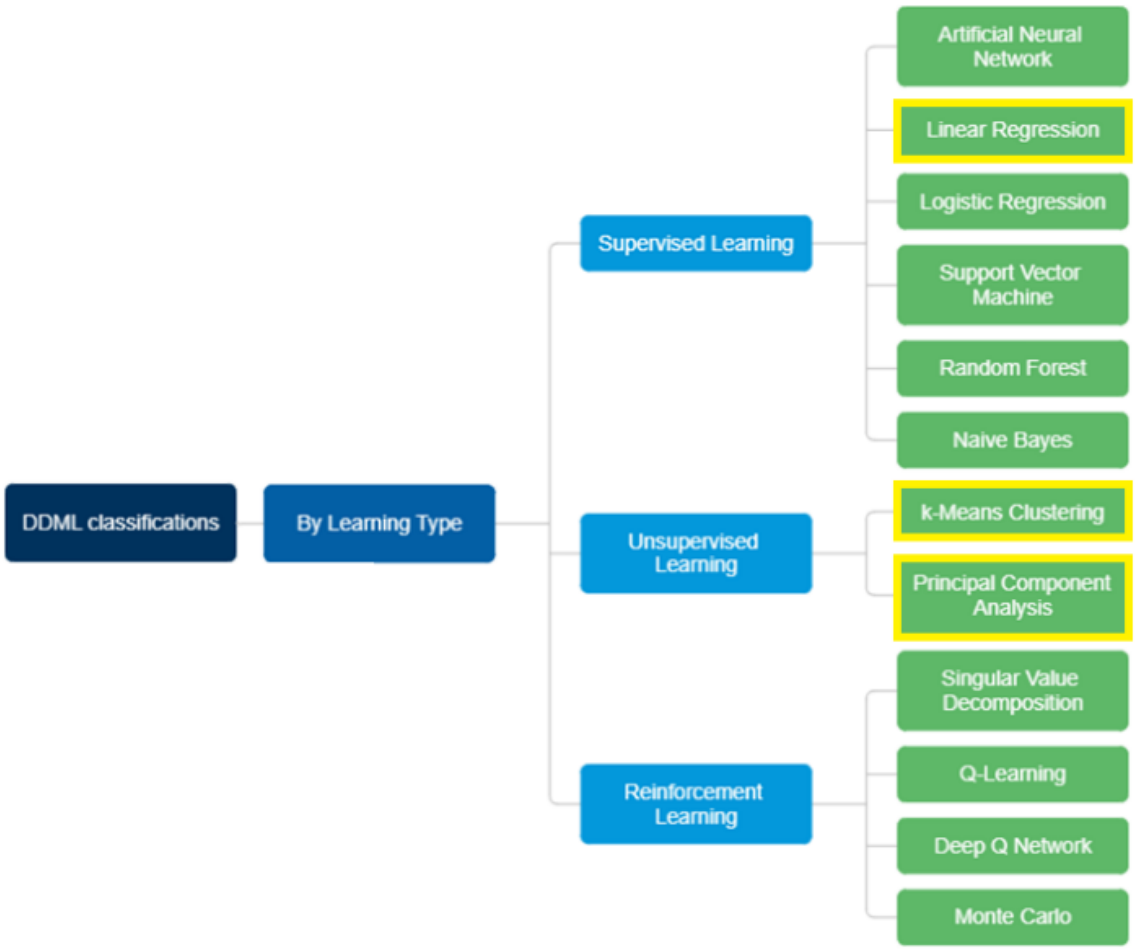


Figure 2. Classifications of DDML, adapted from [49]. This thesis focuses on one method of supervised learning (i.e. linear regression) and two methods of unsupervised learning (k-means clustering and principal component analysis), which are highlighted with yellow borders in the figure.

3.1.1 Unsupervised learning

Unsupervised learning can be categorized based on the approach that is used for the analysis:

- Clustering;
- Anomaly detection;
- Dimensionality reduction.

Clustering involves the segmentation or grouping of features with similar attributes, to better distinguish between possible groups of datasets. It can be used, for example, for

anomaly detection when a data point does not fit in a group. Some examples of clustering algorithms are k-means [50], hierarchical clustering [51], density-based spatial clustering of applications with noise (DBSCAN) [51], ordering points to identify the clustering structure (OPTICS)[52] and multivariate normal distribution [53].

Anomaly detection is used to find data or observations that do not fit in a normal operation of a process. Anomaly detection techniques include isolation forest [54] and local outlier factor [55].

Dimensionality reduction encompasses the transformation of data from a higher dimensional space to a lower-dimensional space. In this process, there should remain relevance of the low-dimensional space, so that it can represent the data of the original high-dimensional space, but there will be losses of data representation during the transformation. Dimensionality reduction techniques include principal component analysis (PCA) [56] and non-negative matrix factorization (NMF) [51].

3.1.1.1 Unsupervised learning method 1: PCA

PCA is a dimensionality reduction technique that performs linear mapping of data to the low-dimensional space. In the process, the variance of data representation in the resulting space is maximized, resulting in new components that represent the original features, without losing relevance to the original information. To perform the PCA, the covariance matrix is constructed and the eigenvectors are computed for this matrix. Based on the eigenvectors that have the largest eigenvalues, the original data can be represented by the recalculation of the matrix using the eigenvectors with the larger eigenvalues. In practice, the first eigenvalue often contributes for the majority of the representation of the behaviour of the system, but it should be validated based on the actual system being analysed. The PCA technique is sensitive to outliers and anomalies in the data.

3.1.1.2 Unsupervised learning method 2: K-means

K-means clustering is a method used for grouping of data by partitioning *n* observations into *k* clusters. Each observation belongs to the cluster with the nearest mean. For the method, there is needed to first define the amount of cluster that the algorithm should categorize the data into. The algorithm defines randomly the centroids, calculates the distance from each point to each centroid, associate the points with the closest centroid and recalculate the centroid positions. This process is run until the centroid positions do

not move. For the distance calculation between the points and the centroids, there is generally used squared Euclidean distance calculation.

3.1.2 Supervised learning

Supervised learning involves training a model using labelled data, i.e. for a known combination of input data, the algorithm should be able to provide an expected output value. Supervised learning can be split into classification (separates the data into specified categories) and regression (models the correlation between input and output parameters) types.

3.1.2.1 Supervised learning method 1: Linear regression

Linear regression is a method of modelling the relationship between dependent and independent variables. The dependent variable is the expected outcome of the modelling and the independent variables are the input values for the model. When there is more than one independent variable, then the process is called multiple linear regression.

The multiple linear regression can be expressed as:

$$y_i = \beta_0 + \beta_1 \cdot x_{i1} + \cdots + \beta_k \cdot x_{ik} + \varepsilon_i, \quad (1)$$

where y_i is the dependent variable, $x_{i1} \ldots x_{ik}$ are the independent variables, $\beta_1 \ldots \beta_k$ are the coefficient parameters, β_0 is the intercept term and ε_i is the error variable. The goal of the linear regression method is to find the coefficients $\beta_0 \ldots \beta_k$ so that the error term is minimized. This process is called fitting of the model.

3.2 Network optimization

A network can be a wired or wireless connection of different devices, used to share data. Network optimization is monitoring, maintaining, and improving the performance of the network, using different tools, techniques and practices. It is usually done by combining a network model and an optimization algorithm. The model is responsible for the prediction of performance for a specific configuration and the algorithm generates the configurations that could meet the expected performance. For network optimization, there should be considered that optimization is possible only for a process or structure that has been modelled. According to [44], the most common network modelling techniques are

i) analytical models, ii) fluid models, and iii) packet-level simulators. A short description of these three techniques is provided in what follows.

The analytical model based on queuing theory is commonly used, but it struggles to accurately model real-life networks with multi-hop routing [44].

Fluid models are popular alternatives, and they are quite simple and useful for several optimization tasks (e.g., link utilization balancing). However, the disadvantage of fluid models is limited accuracy in networks with high utilization regimes and complex queuing policing because these models assume constant per-link delays and do not consider the effects of queuing delays, scheduling policies, network losses [44].

Packet-level simulators are the most accurate in comparison to traditional network models but have a high computational cost and therefore they are not optimal for real-time scenarios with large traffic volumes [44].

To combat these issues, ML-based techniques could provide more effective modelling of networks, by training them on online and offline real-world data from the entire range of the network characteristics.

In Figure 3, adapted from [57], different network optimization objective classes are shown. For this thesis, the focus is on i) QoS parameters, especially bit rate, delay, packet loss, and ii) congestion control, as highlighted in the figure.

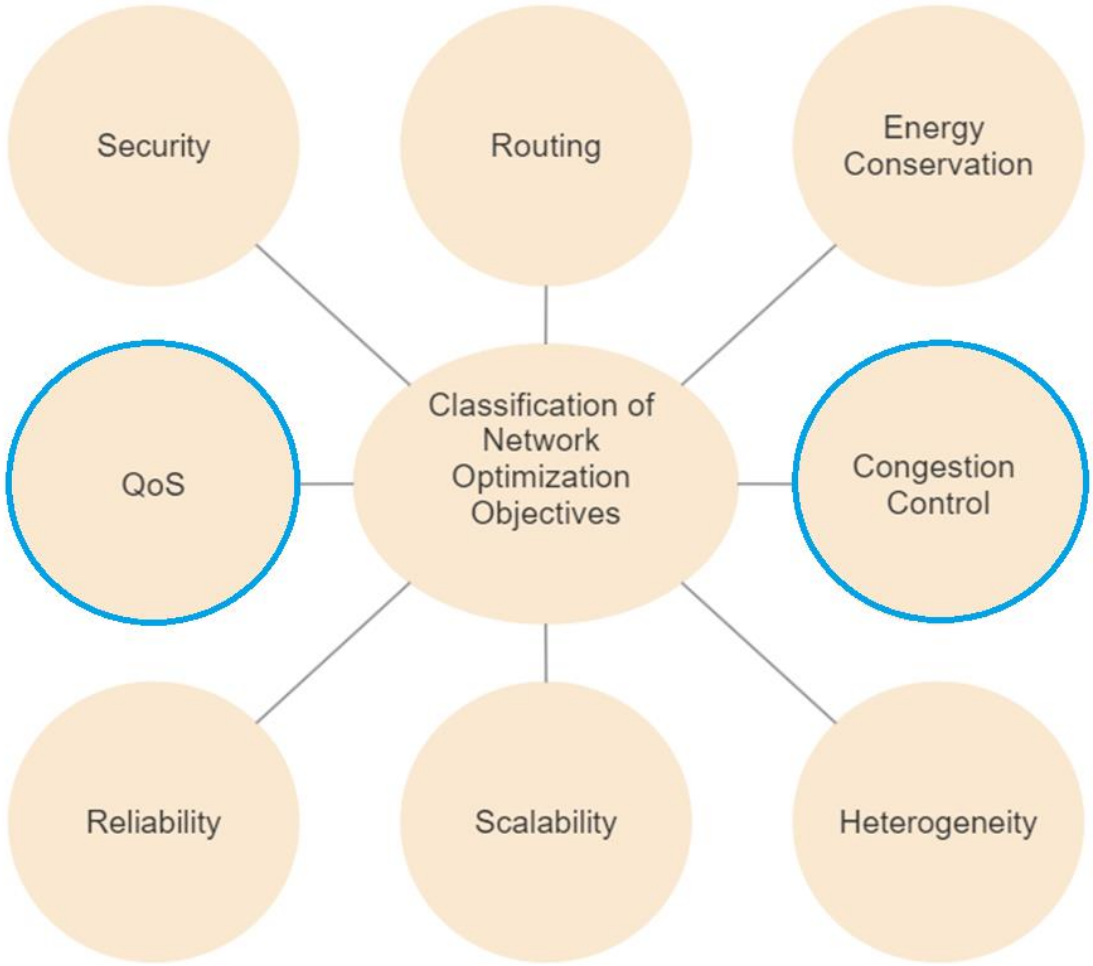


Figure 3. Classification of network optimization objectives for IoT networks, adapted from [57]. As highlighted with blue circles, in this thesis, the focus is on QoS parameters (bit rate, delay, packet loss), and on congestion control.

3.3 Applying ML for network optimization

ML-based network optimization techniques are promising, as they could provide more accurate network modelling by using real-world data for training and additionally could be used to develop a more accurate optimization algorithm. For this reason, the ML algorithm could be utilized in the following two steps:

1) Using unsupervised learning to find relationships in the existing data, perform dimensionality reduction and apply clustering;
2) Using another learning technique to optimize the parameters based on the selected data from Step 1), to provide the control information for the device controlling or optimizing the network parameters.

To be able to select or cluster the data from the available data, the two possible methods are PCA and k-means clustering. PCA (see additional information in Section 3.1.1.1) can help to reduce the number of features used for the calculations and k-means (see additional information in Section 3.1.1.2) clustering can help to define the clusters of data.

Another option is to use supervised learning methods, for example, to determine the relation of the observed data by applying regression modelling techniques (such as linear regression explained in Section 3.1.2.1), to find out if the observed input data correlates to the expected output data.

3.4 Transmission Control Protocol

To improve transferability of the algorithm to be developed, the target for this thesis is to use the trace information obtained from the transport or the layers above it from the network OSI model. One of the most common protocols from the transport layer is TCP. Compared to user datagram protocol (UDP), which is used for time-critical application like streaming or DNS lookups and does not have error correction or packet sequencing, TCP is a protocol which has a goal of making sure the packets have been successfully delivered. TCP uses error correction, packet ordering, retransmissions of lost packets, sliding window flow control and congestion control to improve the reliability of the connection and the delivery of data. For the data to be transferred, the protocol divides it into segments and includes the TCP header to the segment.

Figure 4 shows the TCP header composition, followed by Table 3 which describes selected fields of the header.

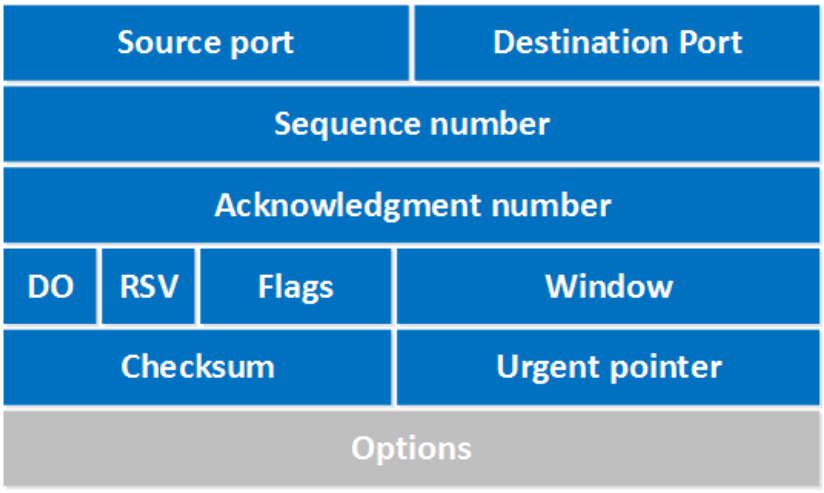


Figure 4. TCP header composition [58].

Table 3. TCP header elements description [58].

Field name	Size (bits)	Description
Source port	16	Sending port ID
Destination port	16	Receiving port ID
Sequence number	32	Initial seq number if SYN flag raised, otherwise accumulated seq number
Acknowledgment number	32	Acknowledges that all previous data has been received
DO	4	Data offset, defines the size of the TCP header
RSV	4	Reserved, not used
Flags	8	Contains the flags: CWR (Congestion window reduced), ECE (Explicit Congestion Notification Echo), URG (Urgent), ACK (Acknowledgment), PSH (Push), RST (Reset), SYN (Synchronize), FIN (Final)
Window	16	Defines size of window that the receiver is willing to receive
Checksum	16	Checksum for the header, data and IP pseudo-header
Urgent pointer	16	Indicates last urgent data byte, if URG flag is raised
Options	0-320	Variable field with different options, size determined by DO

The TCP connection diagrams for normal conditions can be divided into three variants: connection establishment, data transfer, and connection termination. Simplified diagrams of different the three connection variants of TCP [59] are shown in Figure 5.

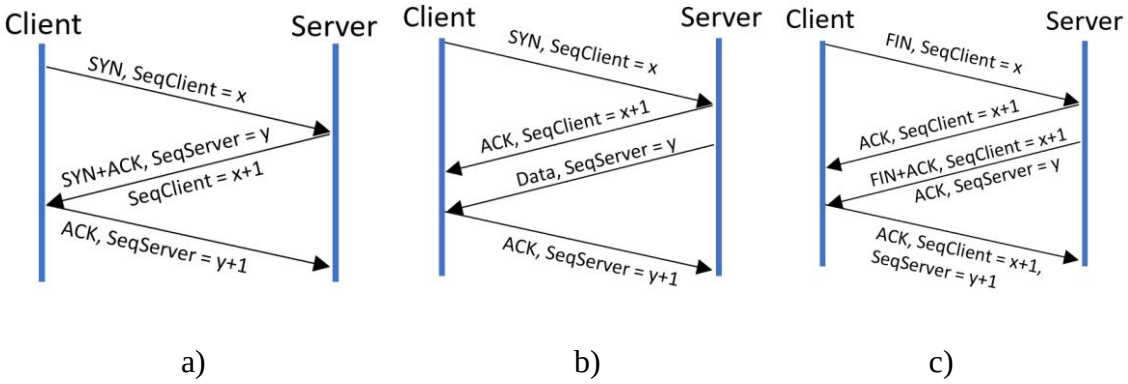


Figure 5. TCP diagrams for connection variants a) connection establishment, b) data transfer, c) connection termination, adapted from [59].

For acquiring and analysing the datasets used in this thesis, the software *Wireshark* and its command line utility *tshark* were used; this enables capturing traces from a network interface of the device it is installed on. The Wireshark software includes additional TCP analysis tools, which provide the following example metrics that can be extracted from a TCP trace file, which were used for the data analysis for this thesis: duplicate ACK packets, suspected retransmission packet and previous segment not captured indicator.

This chapter gave an overview of the background theory for data-driven techniques, network optimization, using data-driven techniques for network optimization and TCP. The next chapter presents the details of the design and implementation of the proposed learning algorithms, including the choice of hardware and the obtaining of the datasets.

4 Design and Implementation of the Proposed Algorithms

This chapter presents the algorithms that have been developed in this thesis, their implementation and the method of acquiring the datasets for analysis and algorithm development.

4.1 Algorithm development

In this thesis, two possibilities of algorithm development have been investigated, i.e. unsupervised learning with PCA and k-means and supervised learning with linear regression.

4.1.1 Unsupervised learning algorithm development

For unsupervised learning, the parameters listed below from the TCP trace files were examined, to determine any noticeable patterns that could be used to determine different network conditions. To create the traces, several constraining conditions were applied to the network which the observed devices were connected to. These constraints are described in Section 4.2.

The datasets were examined using both PCA and k-means clustering, to determine if the dimensionality could be reduced and if the different simulated conditions could be separated from the data.

The parameters and different combinations of these parameters used for the analysis during PCA are as follows:

- Measurement time (s)
- Time between segments for same transmission (s)
- Length of packet (bytes)
- Length of data (bytes)
- Size of packet (bytes)
- Window size (bytes)
- Relative acknowledgement number
- Timestamp value
- Timestamp echo reply value
- Number of suspected retransmission packets

- Number of duplicate ACK packets
- Number of suspected lost packets

For applying the k-means clustering method, the same parameters were used as for PCA.

Figure 6 shows the proposed flow-chart of the unsupervised learning algorithm, containing the steps 10 to 110.

The Step 10 involves the initialization of the variables and constants. The Steps 20, 30 and 40 correspond to the capturing of the trace data, converting it to .csv and opening it as a dataframe correspondingly. The Step 50 involves processing the data, transforming it into a suitable format for the algorithm. In Step 60, the PCA is applied to transform the data into suitable principal components. In Step 70, k-means clustering is applied to the transformed PCA data. In Step 80, the output of the algorithm is calculated based on the clustering result. In Step 90, the result is compared to the previous iteration of the algorithm. If the result is not changed, the algorithm goes to Step 110 to save the result in a log file and start a new cycle from Step 20. If the algorithm outcome has changed, then the Step 100 is executed, which includes updating of the network parameters to optimize the network.

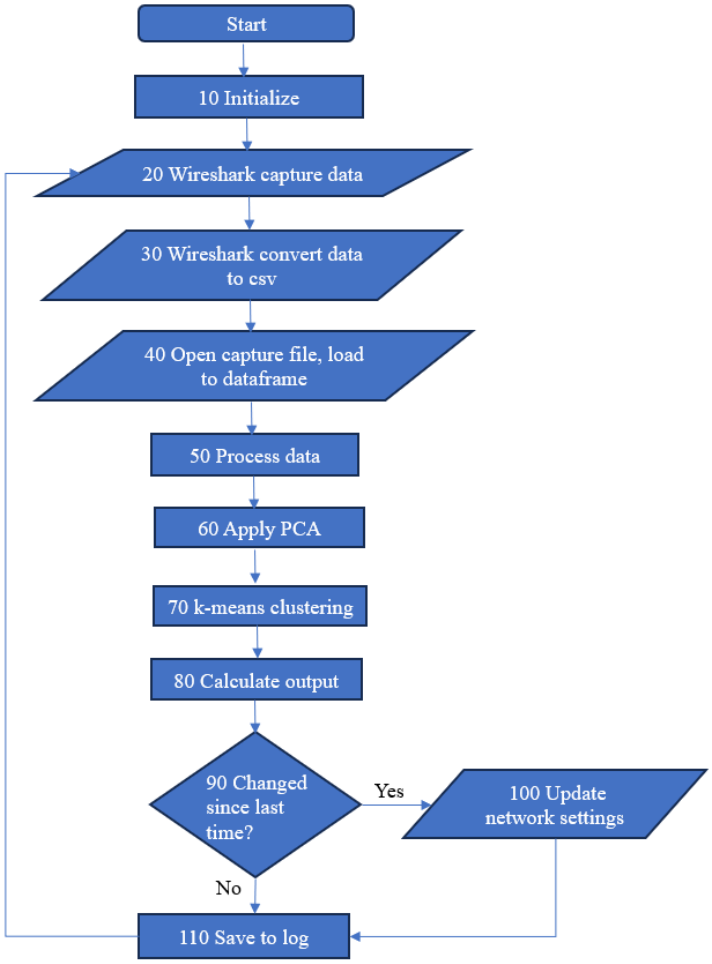


Figure 6. Flow-chart of the proposed unsupervised learning algorithm.

4.1.2 Supervised learning algorithm development

For the second approach, a linear regression analysis was carried out to examine the relationship expressed in Equation (2).

$$F = f(C, T_a, T_m, L, D, R), \tag{2}$$

where *F* is the number of devices connected to the observed network, *C* is the count of measured packets, *Ta* is the Average of RTT, *Tm* is the Median of RTT, *L* is the number of suspected dropped packets, *D* is the number of duplicate ACKs, and *R* is the number of suspected retransmission packets. All of these parameters should be considered to be monitored over a time period, to have more stable results.

When applying the linear regression formula, an ideal case would be as per Equation 3.

$$F = \beta_0 + C \cdot \beta_1 + T_a \cdot \beta_2 + T_m \cdot \beta_3 + L \cdot \beta_4 + D \cdot \beta_5 + R \cdot \beta_6. \tag{3}$$

To execute the algorithm onto the RPi, it would be needed to periodically capture the data, analyze it, and provide an output to update the necessary network parameters. Figure 7 shows the proposed algorithm flow graph, comprising of Step 10 to Step 90.

Initially, in Step 10, the constants and variables are initialized. During Step 20, Wireshark CLI utility tool tshark is used to capture specific TCP packet data during a specified time window and saved as a temporary *pcap* (packet capture file, which stores network capture data, which can later be used to analyze network parameters and troubleshoot for potential issues) file. In Step 30, the data collected in Step 20 is saved as a temporary csv file. Step 40 involves opening the csv file from Step 30 as a *pandas* dataframe (two-dimensional table, with a variable size [60]). Step 50 processes the data in the dataframe, sorting out relevant data and replacing empty rows with corresponding data. Step 60 includes the calculation of the selected values and additionally the regression output value. Step 70 checks if the predicted number of devices has been changed compared to the previous iteration and runs either Step 80, which updates the network parameters correspondingly or Step 90, which saves the required data to a log file and starts a new cycle.

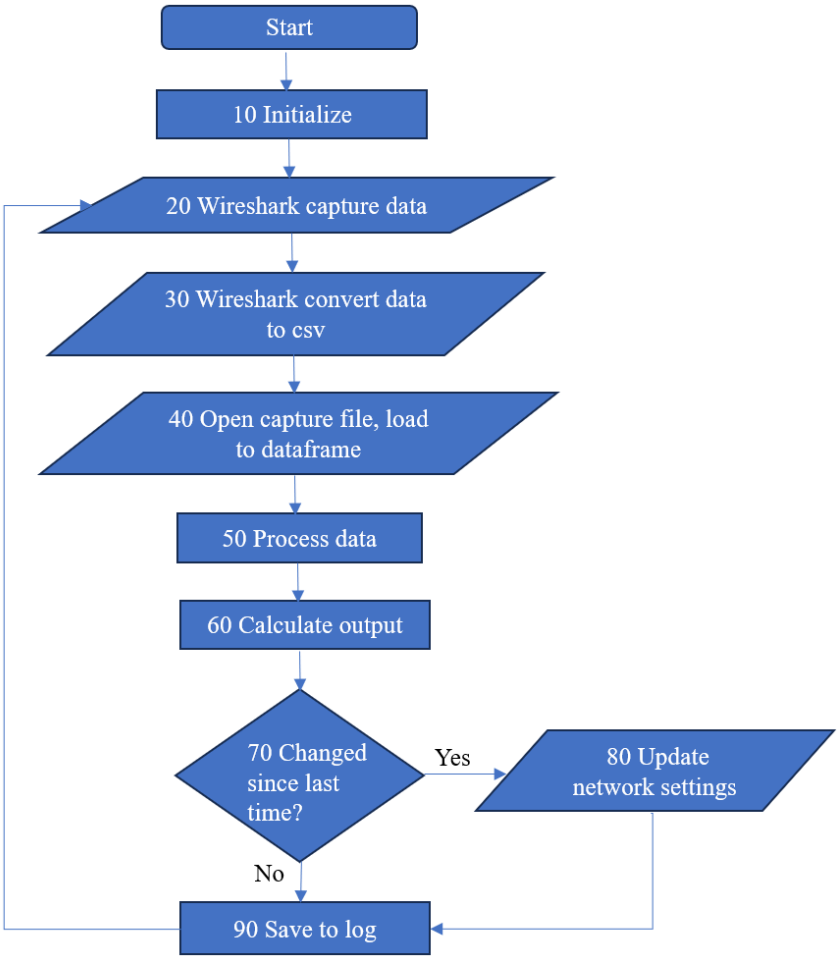


Figure 7. Flow chart of the proposed algorithm with linear regression. Details of the steps are provided in text.

4.2 Dataset

The dataset that was chosen for analysing the problem was collected using a real-world setup with different constraints to the network applied. For the data collection, all connected devices had the same software (SW) for simulating the network traffic programmed. The five main steps of the data collection system flow are shown in Figure 8.

Figure 8. The five main steps of the data collection system flow.

The different constraints that were simulated:

- Bandwidth limitation, ranging from not limited to 10 kbit/s;
- Network loss, ranging from 0% to 25%;
- Network delay for packets leaving the ethernet interface, ranging from 0 ms to 500 ms.

The constraints were partially inspired by recommendations presented in International Telecommunication Union document Y.1541 [61]. The values of these parameters were obtained by using Wireshark to monitor network traffic TCP packets.

4.3 Hardware

Hardware selection involves two parts: i) hardware for training, and ii) hardware for deployment.

For training, the device should have high computational power, which is why Google Colaboratory was chosen for this task. At the time of performing the analysis, there was available the GPU NVIDIA Tesla T4 (with 16 GiB GDDR6) and CPU Intel Xeon with Google Colaboratory entry tier.

For the deployment hardware, there were different SBC-s available for the AP device, which are listed in Table 4.

Table 4. List of SBC-s available for algorithm deployment on the AP device.

Device name	Processor	Memory	Price	Network connectivity
Raspberry Pi 4 Model B [62]	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1,5 GHz	1 GiB, 2 GiB, 4 GiB or 8 GiB LPDDR4	€31,62	2,4 GHz and 5,0 GHz IEEE 802.11ac, Gigabit Ethernet
NVIDIA Jetson Nano [63]	ARM® Cortex® -A57 MPCore (Quad-Core) | 1,43 GHz	Dual Channel | LPDDR4 | 1600 MHz | 25,6 GB/s |4 GiB	€89,44	Gigabit Ethernet
Google Coral DevBoard [64]	NXP i.MX 8M SoC (quad Cortex-A53, Cortex-M4F)	1 or 4 GiB LPDDR4	€117,46	Gigabit Ethernet port, Wi-Fi 2x2 MIMO (802.11b/g/n/ac 2,4/5 GHz)
Raspberry Pi Zero W [65]	1 GHz, single-core CPU	512 MiB RAM	€13,55	802.11 b/g/n wireless LAN

For the reasons of availability, accessibility, price, computational power, the Raspberry Pi 4 model B [62] with 4 GiB of RAM has been selected for the initial testing and validation of the algorithm on the AP device. Figure 9 shows a photograph of this device.

Figure 9. Raspberry Pi 4 model B [66] used for hardware deployment of the algorithm on the AP device.

For the simulation of the network client devices, Raspberry Pi Pico W [67] units were used to generate network traffic. The specifications for the RPi Pico W are as follows: Microcontroller RP2040, dual-core ARM Cortex-M0+ processor, 254 kiB on-chip SRAM, 2 MiB on-board flash, 2,4 GHz 802.11n wireless LAN. Figure 10 shows a photograph of this device.

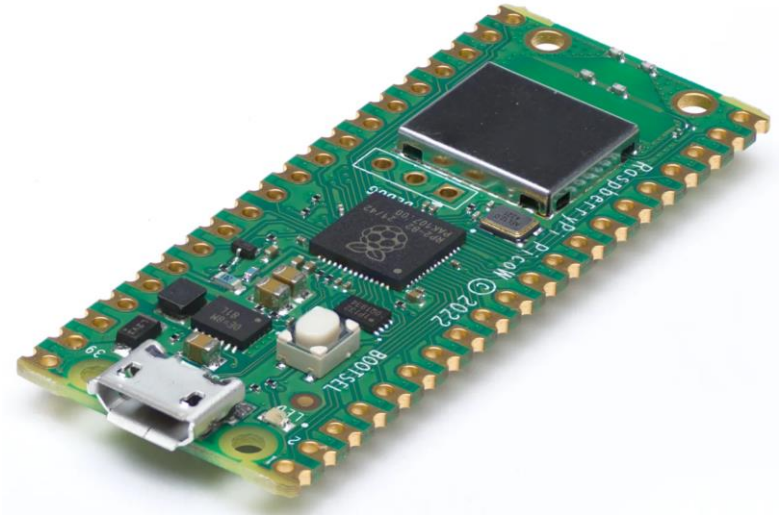


Figure 10. Raspberry Pi Pico W [68] used as network clients.

The structure of the measurements' setup is shown in Figure 11. The number of RPi Pico W devices connected to the RPi 4 can be changed from 1 to *n* (as will be shown later on, *n* is limited to 5 in this setup).

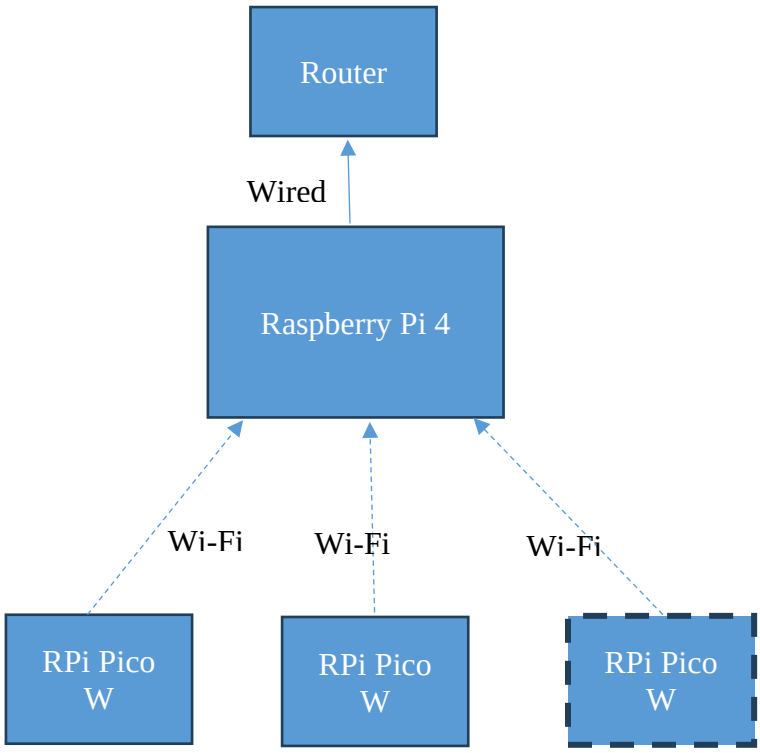


Figure 11. Measurement and validation setup. The RPI Pico W devices act as the network clients and generate the network traffic. They are connected, over WiFi (802.11n 2,4 GHz), to the RPI4 which acts as the AP and implements the proposed algorithm. The RPI4 is connected over a wired Ethernet (Gigabit Ethernet) link to the router.

Figure 12 shows a photograph of the real-life setup of the network corresponding to Figure 11, with 10 RPi Pico W boards used as network clients.

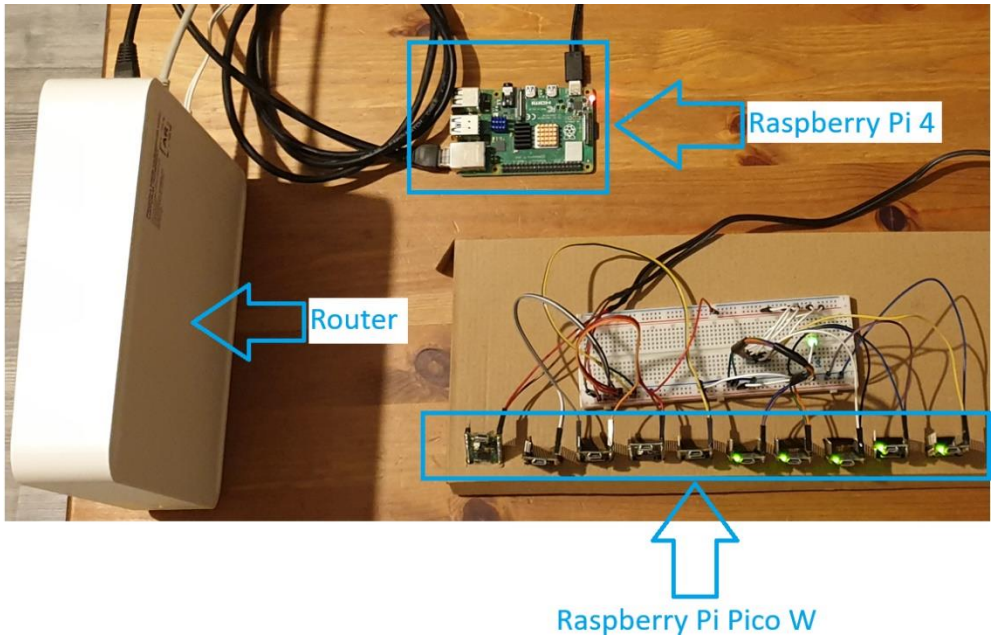


Figure 12. Photograph of the network measurement and validation system setup, corresponding to the diagram shown in Figure 11 As in Figure 11, the RPI Pico W devices act as the network clients and generate the network traffic. They are connected, over WiFi (802.11n 2,4 GHz), to the RPI4 which acts as the AP and implements the proposed algorithm. The RPI4 is connected over a wired Ethernet (Gigabit ethernet connections with supplied maximum bandwidth of 100 Mbit/s) link to the router.

This chapter gave an overview of the design and implementation of the proposed learning algorithms, including the choice of hardware and the obtaining of the datasets. The next chapter presents the details of the results obtained while using different algorithms and the analysis of the results.

5 Results and Analysis

The measurements and validation were performed with a RPi 4B 4 GiB model, running Ubuntu Desktop 64bit OS version 23.10 and Wireshark version 4.0.8. The results are divided into two parts: i) results for the unsupervised learning and ii) results for supervised learning. The unsupervised learning approach presented in Section 5.1 did not provide enough convincing results to proceed with this method, and therefore it was decided to focus on the supervised learning approach; the results for the supervised learning approach are presented in Section 5.2 and are according to expectations. Moreover, Section 5.3 provides an overview of different observed classification parameters (memory usage, time delay).

5.1 Unsupervised learning approach (PCA and k-means)

To determine the suitable number of principal components for the PCA, the scree plot method was applied; it is a graphical method that uses the "elbow" of the graph to determine the number of PCA principal components necessary for the analysis. Figure 13 shows an example scree plot, which suggests that for this example the number of features to be used is approximately 4 (i.e. the Eigenvalues of components 1 to 4 are above the value '1', whereas the Eigenvalues of the components 5 to 10 are below '1'). The scree plot was made using a dataset that combines periodically applied constraints to the network, as suggested in Section 4.2, with non-constrained and each constrained scenario applied in 25% of the observation time.

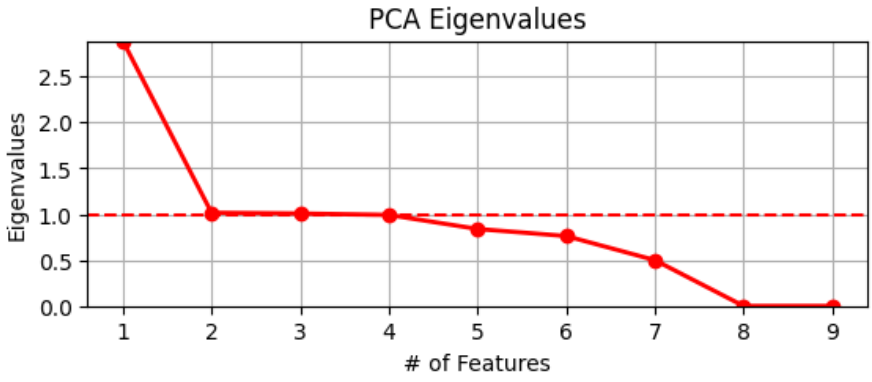


Figure 13. Example scree plot for PCA which suggests that for this example the number of features to be used is approximately 4.

Figure 14 shows the PCA plot for the first two principal components. From the figure, it was found that there are no distinguishable groups that could be used for further analysis.

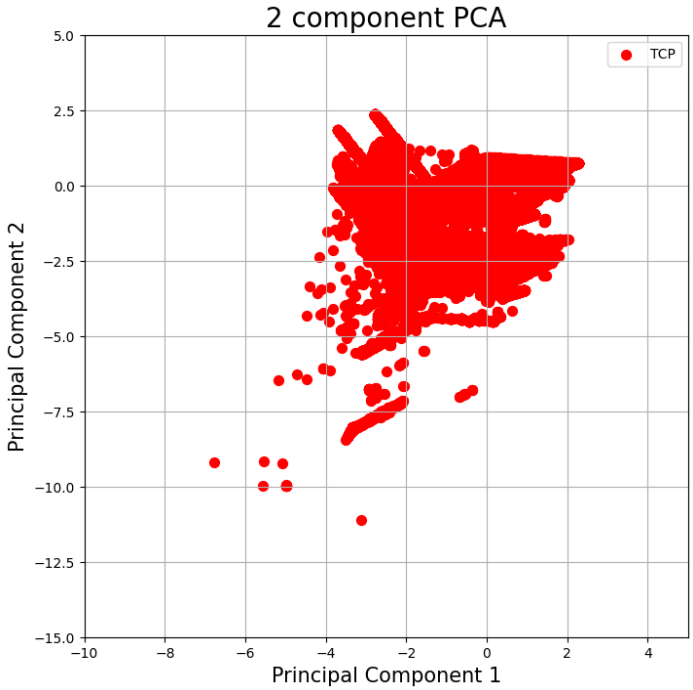


Figure 14. Example two component PCA plot. As can be seen, it was found that there are no distinguishable groups that could be used for further analysis.

Table 5 shows a list of principal components, with their corresponding variance ratio displayed. Table 6 shows a list of principal components with their corresponding explained variance ratio for each initial parameter. The results show that the combined first four principal components have a low variance ratio (0,28 + 0,12 + 0,09 + 0,09 = 0,58, i.e. 58%), when actually it would be desirable to have at least 80% variance explained by these four components. Furthermore, for the main principal components, there is no suggestion that there is a major parameter that is corresponding to the majority of their variance.

Table 5. Variance ratio of principal components for a combined dataset, which includes added delay and bandwidth constraints.

Principal component	Variance ratio
PC_1	0,28
PC_2	0,12
PC_3	0,09
PC_4	0,09

Table 6. Principal components' explained variance ratio for each parameter for combined dataset. TBP – Time between packets, Ret – Suspected retransmission packets, Dup – Duplicate ACK-s, Lost – Suspected lost packets.

	TBP	Length	Seq	Ack	Win	Len	Tsval	Tsecr	Ret	Dup	Lost
PC_1	0,04	0,53	0,29	0,22	0,20	0,53	0,38	0,31	0,03	0,15	0,01
PC_2	0,08	0,04	0,12	0,38	0,61	0,05	0,28	0,51	0,33	0,08	0,01
PC_3	0,68	0,02	0,01	0,02	0,10	0,02	0,00	0,12	0,35	0,53	0,33
PC_4	0,12	0,02	0,02	0,08	0,04	0,02	0,04	0,04	0,32	0,19	0,92

Next, to determine the number of clusters to use for the k-means clustering, the elbow method was also used and the result is displayed in Figure 15. The figure suggests that the number of clusters should be 7. But as the number of applied known scenarios was 4, then this method was also deemed not suitable for further investigation.

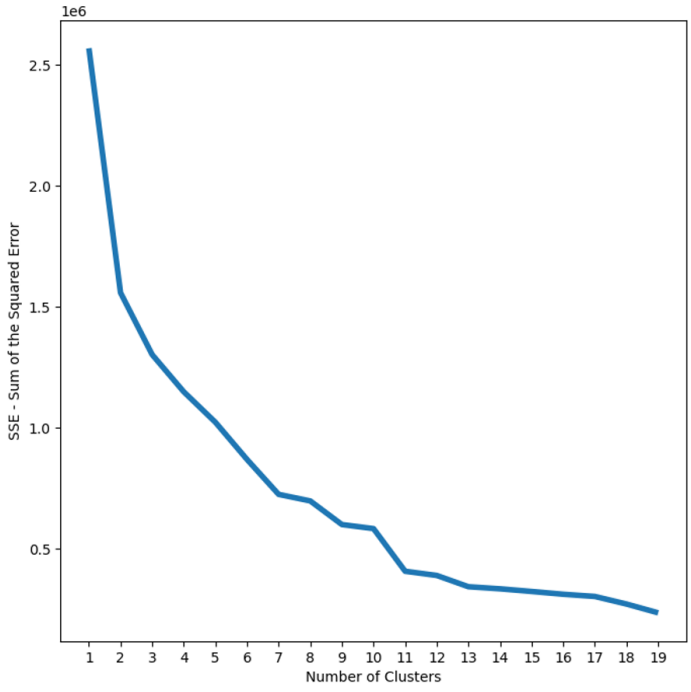


Figure 15. Elbow curve for k-means for a combined dataset case.

The results of the above investigations show that the dimensionality cannot be significantly reduced, because it was found that the principal components do not have significant variance ratios and there is no clear parameter that has a weight that is significant enough. Additionally, it was also found that there are no clearly distinguishable clusters that could be distinguished to correspond to the scenarios expressed by the dataset. Appendix 1 provides additional results for the unsupervised learning approach, which consist of further combinations of input parameters and number of components used for the PCA, but it was also found that these results do not provide a clear conclusion. Appendixes 6 and 7 contains the source codes for the PCA and k-means algorithms, respectively, that were used to analyse the datasets.

Due to these results, despite the efforts spent in trying various combinations, the unsupervised learning approach with the PCA and k-means techniques was not pursued any further in this thesis. Instead, the focus was changed to a supervised learning approach, i.e. linear regression analysis, presented in what follows.

5.2 Supervised learning

During capturing of the different datasets, it was discovered that the hardware and software combination only supported the connection and monitoring of five network client devices[2] out of the 10 available client devices. Due to this technical limitation, the datasets were collected for one to five devices connected to the network with different applied constraints. The constraints were applied using the *netem* tool, which is a Linux network emulation tool that provides functionality to emulate different network conditions (bandwidth, delay, loss, jitter, packet reordering). At first, the datasets were analysed for each parameter separately, to see which parameters had larger correlation to the number of devices connected to the network. The analysis was done with sampling the data at different time windows, to observe its impact on data stability. The source code

[2] The constraint was discovered during experimental activities and there was no indication beforehand that this would become an issue. Research in the RPI community fora shows that there might be an upper limit to the number of devices connected, but it should not be as low as five; at the time of writing, this issue remains open.

for calculating the regression coefficients and the coefficient of determination is displayed in Appendix 7.

The results are shown in Table 7. The results show that the count of measured packets, median of RTT, and number of suspected retransmission packets had the highest correlation to the number of devices connected to the network.

Table 7. Regression coefficients for different parameters for several network constraint scenarios. Network traces captured with 1 to 5 client devices connected.

	No constraints	10 kbit/s bandwidth constraint	25% added loss	25% loss and non-constrained	All datasets (non-constrained, 25% additional loss, 10 kbit/s bandwidth constraint) combined
Count of measured packets	1	0,44	0,52	0,74	0,63
Average of RTT	0,42	0,83	0,02	0,16	0,05
Median of RTT	0,81	0,59	0,01	0,25	0,13
Count of duplicate packets	0,02	0,09	0,44	0,16	0,14
Count of suspected retransmission packets	0,32	0,09	0,62	0,46	0,3

Regarding the observation time window, there were examined options from 1-30 s (the time windows were chosen empirically, as suitable references could not be found in the literature). It was determined from the initial analysis that a time window from 5-15 s would achieve a suitable compromise between model accuracy and how fast the device would periodically produce output from the algorithm.

Figure 16 shows a summary of this analysis. Although in the PCA analysis it is desirable to have 80% of explained variance from the principal components, it is not applicable to linear regression coefficient of determination and therefore it was decided that a coefficient of determination lower than 80% would also be suitable for linear regression. Furthermore, there were no concrete requirements found in the literature for the suggested value of this parameter, as it depends on the system and field of study analysed.

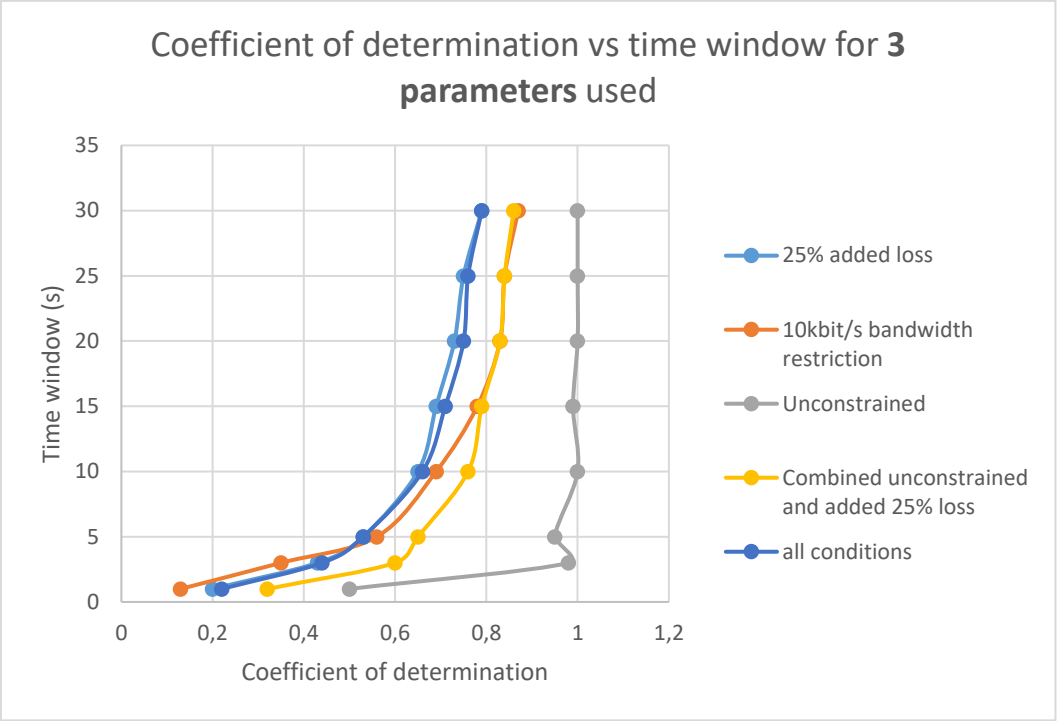


Figure 16. Coefficient of determination vs time window for 3 parameters used for regression analysis for different conditions. It can be seen in the graphs that the slope of the graphs starts to even out after 5 - 15 s, which is why this region was focused on in the development of the algorithm.

Figure 17, Figure 18, and Figure 19 show the result of the regression analysis with a time window of 10 s for the parameters that had the highest correlation factor – normalized count of measurements, median of RTT and normalized count of retransmission packets, respectively. Appendix 5 presents additional regression results for other capture time windows (5 s, 15 s) and additional parameters for a 10 s time window (average of RTT, count of duplicate packets) for a combined dataset.

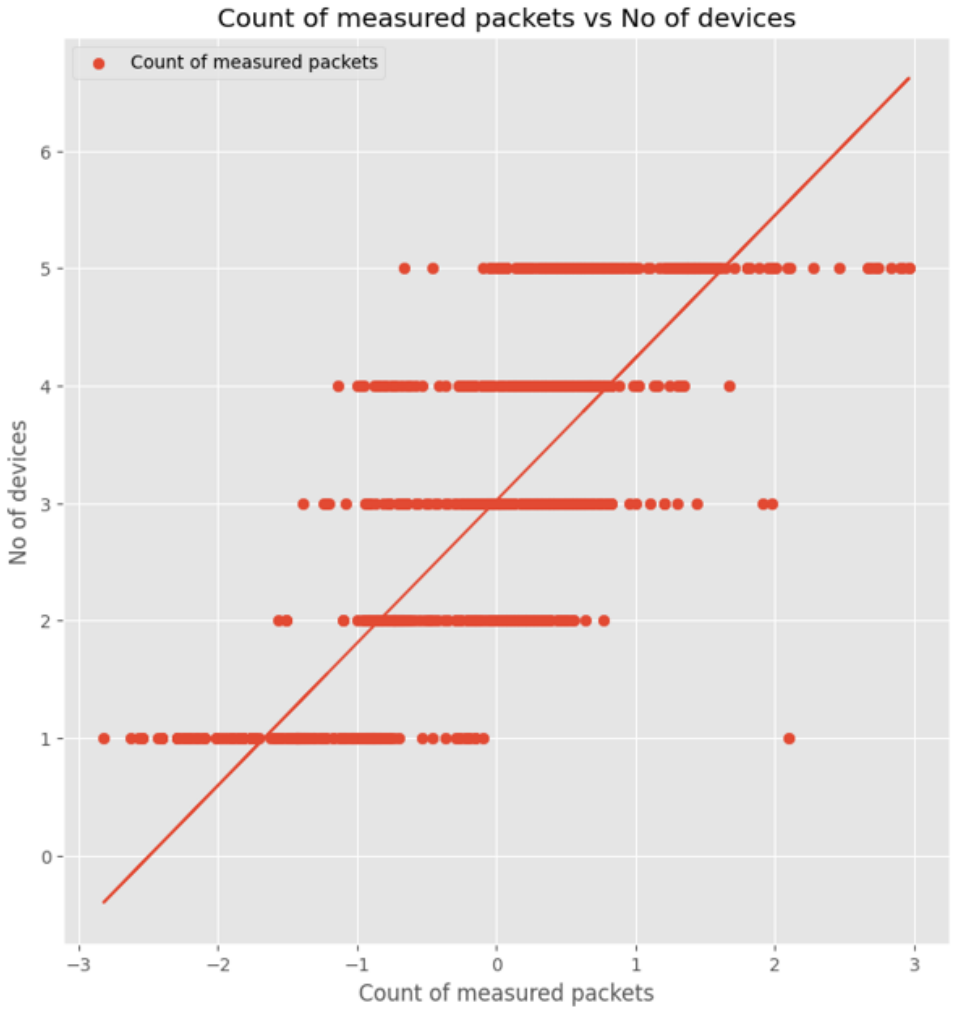


Figure 17. Count of measured packets vs number of devices, for a capturing time window of 10 s.

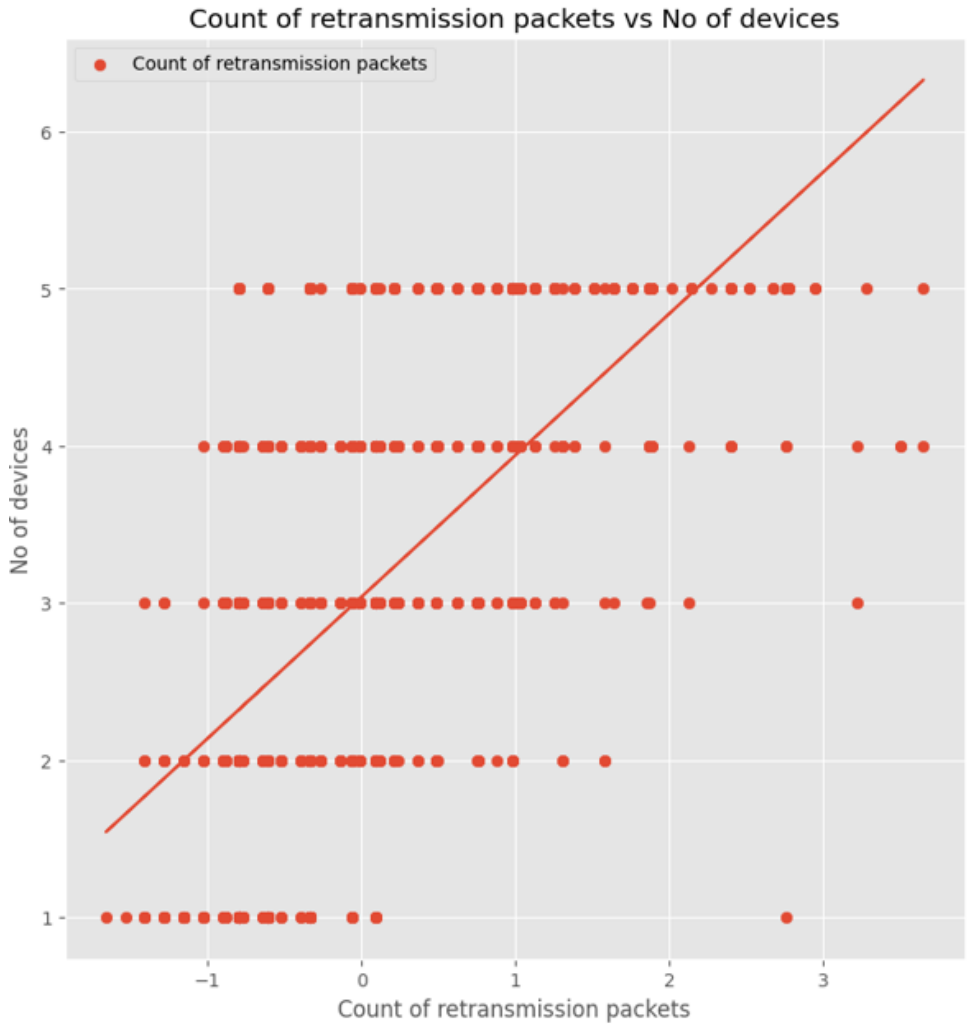


Figure 18. Count of retransmission packets vs number of devices, for a capturing time window of 10 s.

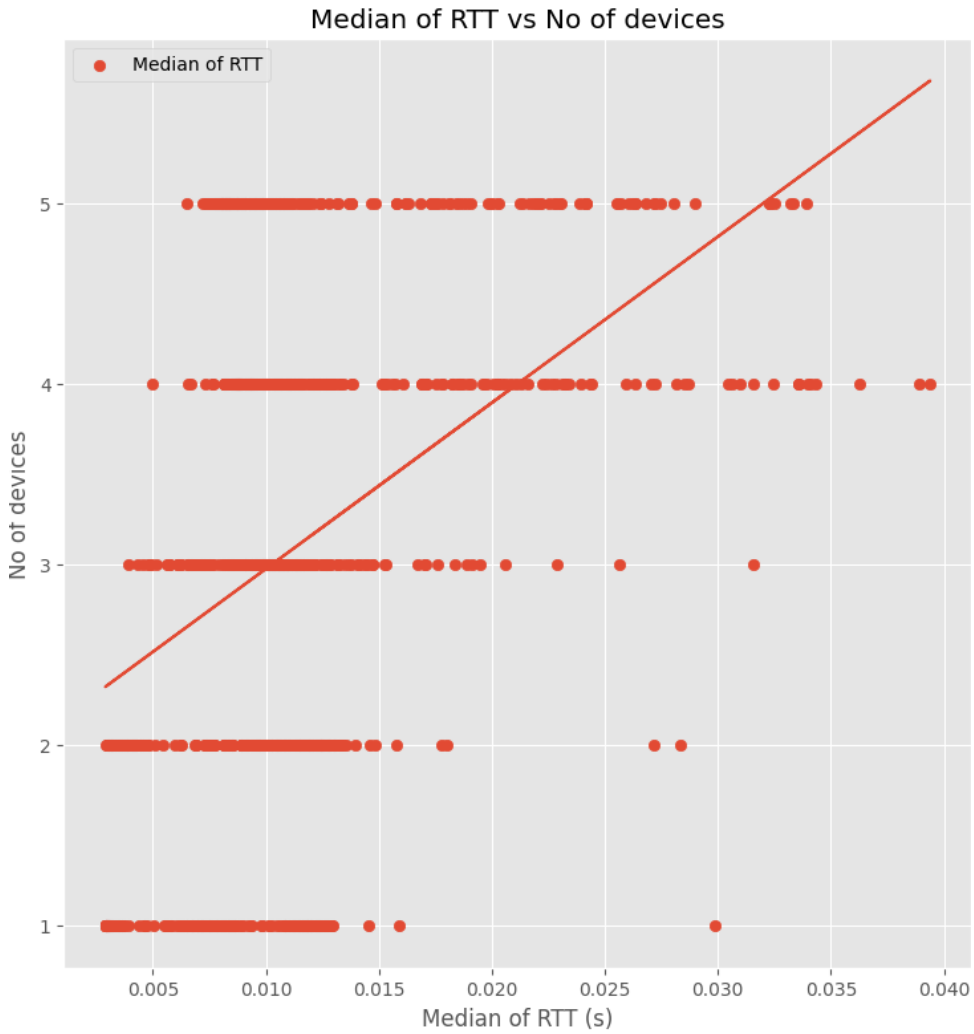


Figure 19. Median of RTT vs number of devices, for a capturing time window of 10 s.

In ML it is advisable to split the dataset into training, validation, and testing sub-datasets, to check the accuracy of the model being trained [69]. But as the *sklearn* library used for linear regression calculation does not have hyperparameters to tune, the datasets were also examined with the k-fold cross validation method, to see how the error rate would differ when selecting different sections of the dataset for analysis. For the k-fold cross validation, 10 folds were used, which means that the dataset was split into 10 folds (sub-datasets) and for each of those 10 folds, the parameters of the model were calculated using that part of the dataset. Table 8 presents the results for this analysis.

Table 8. k-fold cross validation results for different datasets. MAE – Mean absolute error. RMSE – Root mean square error. R^2 – Coefficient of determination.

Dataset used	Average MAE of 10 folds	Average RMSE of 10 folds	R^2 across whole dataset	Average R^2 of 10 folds
Unconstrained	0,05	0,0052	0,9975	0,9973
Combined	0,6424	0,6895	0,6554	0,6476

The analysis was also performed by splitting the dataset into 80%-20% (training-testing) and 80%-10%-10% (training-validation-testing), to verify whether there would be a significant difference in the results. The results showed that there was no significant difference in the parameters (unconstrained dataset training MAE varying from 0,0503 to 0,0628, RMSE from 0,0733 to 0,1185, R^2 from 0,9974 to 0,9975 when using the 80%-20% and 80%-10%-10% split method, respectively) and based on this it was decided to use the whole dataset for the linear regression algorithm training.

To determine whether it would be possible to further reduce the number of parameters used, an additional PCA was performed for these latest parameters. The results for this experiment are provided in Appendix 4. In the results, there appear to be clusters corresponding to the expected output of the algorithm, but there is also additional noise in the resulting plots, so this additional PCA possibility was not pursued any further.

For deploying the algorithm to the device, capture windows of 5 ,10 and 15 s were used.

Based on these results, the linear regression was finalized as per Equation 4.

$$F = \beta_0 + C \cdot \beta_1 + T_m \cdot \beta_3 + R \cdot \beta_6, \quad (4)$$

Table 9 shows the regression parameters for 5, 10 and 15 s capture time windows.

Table 9. Correlation parameters for 5, 10, 15 s capture time windows.

Capture time window (s)	β_0	β_1	β_2	β_3
5	-0,4	0,0028	90,4	-0,0008
10	-1,04	0,0039	280,7	0,1111
15	0,3	0,0021	-20,2	0,0299

In Table 10, the coefficients of determination for 5, 10 and 15 s capture time windows are shown, for the combined dataset.

Table 10. Coefficients of determination for 5, 10, 15 s capture time windows, for the combined dataset.

Capture time window (s)	Coefficient of determination
5	0,53
10	0,66
15	0,71

The model was deployed and run on the AP device with different number of client devices connected to the network. The source code for the algorithm that was run on the AP device can be seen in Appendix 2. The source code that was run on the client devices is displayed in Appendix 3. Figure 20 is an example of the output log file.

```
13-12-2023, 02:27:50   Number of connected devices - 2
13-12-2023, 02:28:16   Number of connected devices - 2
13-12-2023, 02:28:42   Number of connected devices - 2
13-12-2023, 02:29:08   Number of connected devices - 2
13-12-2023, 02:29:34   Number of connected devices - 2
13-12-2023, 02:30:00   Number of connected devices - 2
13-12-2023, 02:30:26   Number of connected devices - 2
13-12-2023, 02:30:52   Number of connected devices - 2
13-12-2023, 02:31:18   Number of connected devices - 2
13-12-2023, 02:32:13   Number of connected devices - 2
13-12-2023, 02:56:00   Number of connected devices - 1
13-12-2023, 02:56:28   Number of connected devices - 1
13-12-2023, 02:56:56   Number of connected devices - 1
13-12-2023, 02:57:24   Number of connected devices - 2
13-12-2023, 02:57:51   Number of connected devices - 2
13-12-2023, 02:58:19   Number of connected devices - 2
13-12-2023, 21:02:17   Number of connected devices - 1
13-12-2023, 21:02:45   Number of connected devices - 1
```

Figure 20. Algorithm output log file.

Table 11 shows the results for the algorithm accuracy for the 5, 10, 15 s capture time windows, when using a non-constrained network setup.

Table 11. Algorithm mean accuracy for 5, 10, 15 s capture window, for non-constrained network setup.

Capture time window (s)	Algorithm mean accuracy
5	29,75%
10	94,67%
15	93,62%

The accuracy was calculated by dividing the number of correct predictions by the total number of measurements. The accuracy is acceptable to be used for further development.

Figure 21, Figure 22, Figure 23 plot the algorithm number of devices estimation results for 5, 10 and 15 second capturing time windows, respectively.

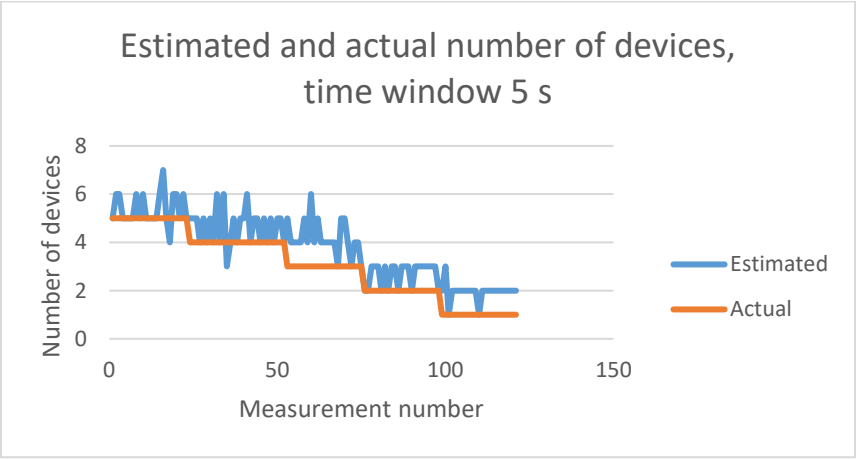


Figure 21. Estimated vs actual number of devices, time window 5 seconds. Model mean accuracy is 29,75%.

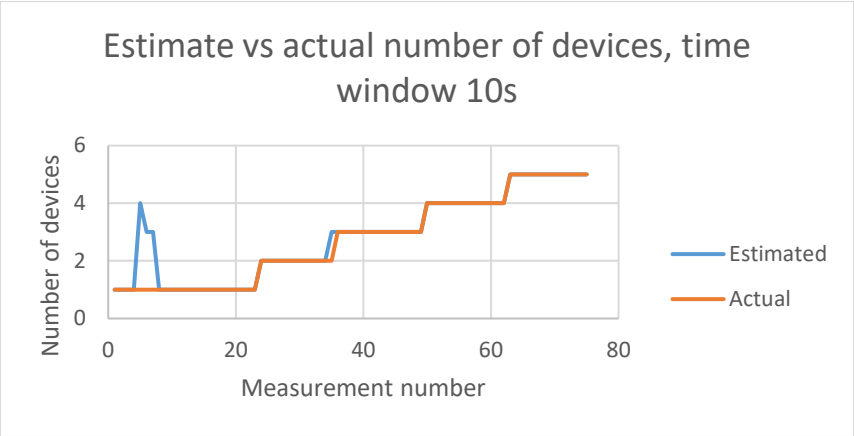


Figure 22. Estimated vs actual number of devices, time window 10 seconds. Model mean accuracy is 94,67%.

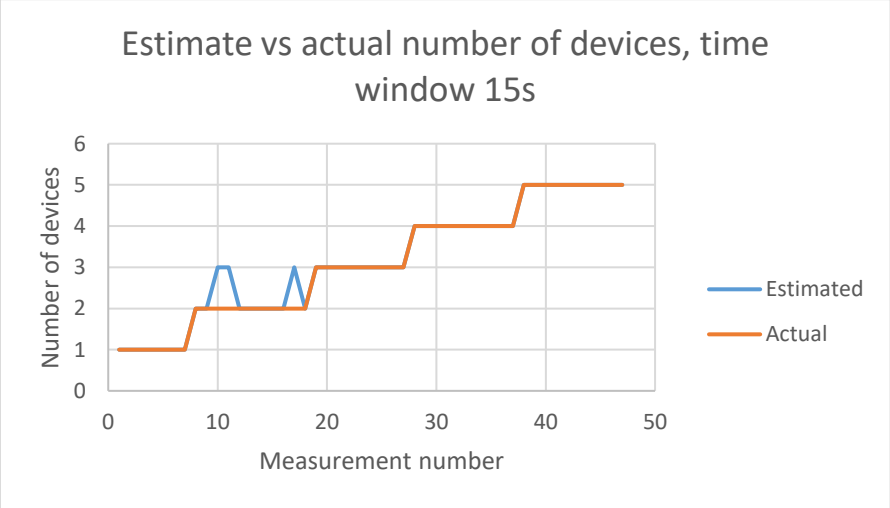


Figure 23. Estimated vs actual number of devices, time window 15 seconds. Model mean accuracy is 93,62%.

Testing with different amounts of data, the results show that when randomly decreasing the dataset size, then after a certain amount of data, the accuracy of the algorithm starts to decrease significantly. On the other hand, it was observed that the minimal amount of data required for acceptable algorithm performance depends on the dataset used, which means that it might be affected by the variation expressed in the data. Figure 24 shows the amount of data used compared to the accuracy of the algorithm for an unconstrained dataset, where the accuracy of the algorithm starts to decrease significantly when the amount of data is below 10 datapoints. For this experiment, the dataset was split into training and testing datasets with a ratio of 80%-20% and the training dataset would be

reduced, while the testing dataset would be used for the accuracy estimations by using the algorithm coefficients calculated from the training dataset.

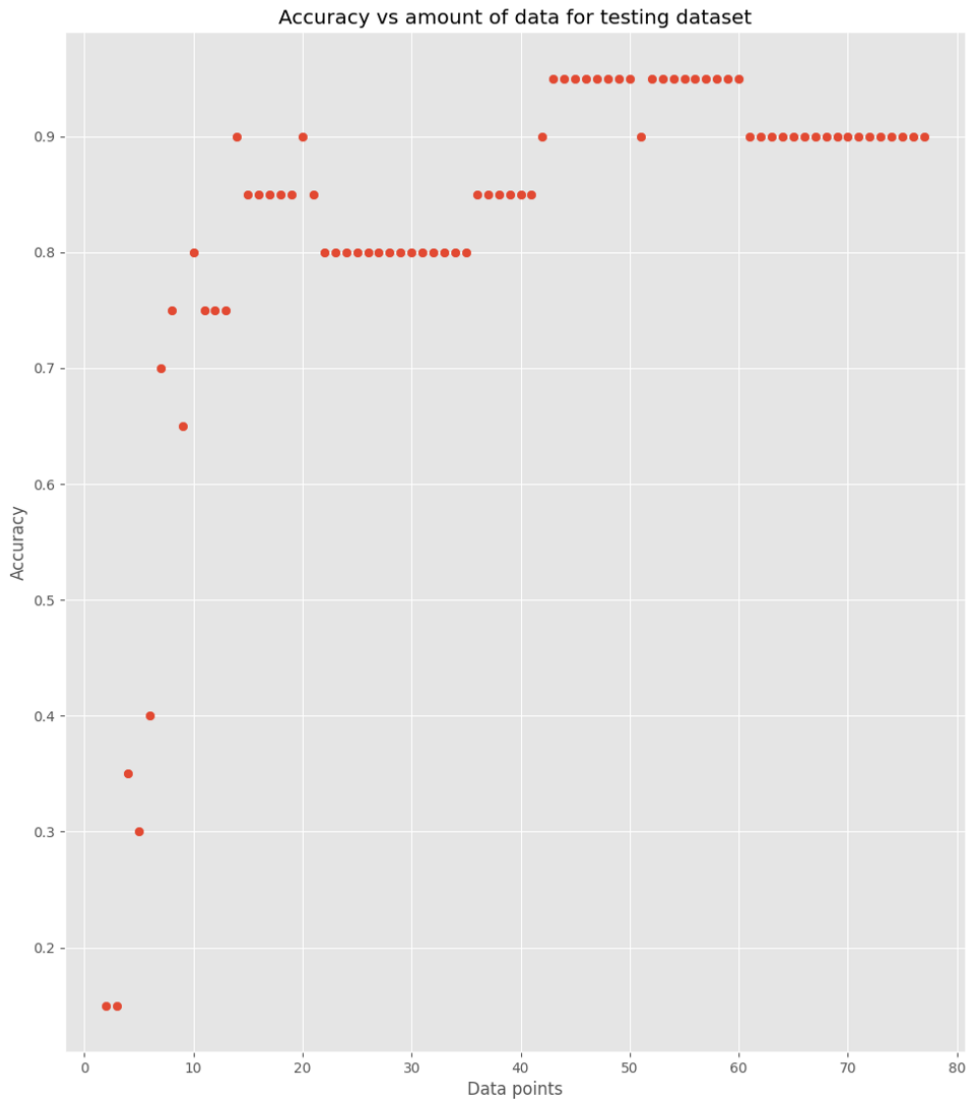


Figure 24. Accuracy vs amount of data used for the linear regression algorithm, using an unconstrained dataset.

While the proposed linear regression-based algorithm provides suitable results in terms of accuracy, it is also important to consider its implementation cost in terms of memory and execution time usage, which is discussed in what follows.

5.3 Linear regression-based algorithm memory and time usage classification

For the supervised learning classification estimation, both time and memory usage estimation tests have been performed. Figure 25 shows the execution time delay (summation of the data processing and of the result calculation) of the algorithm.

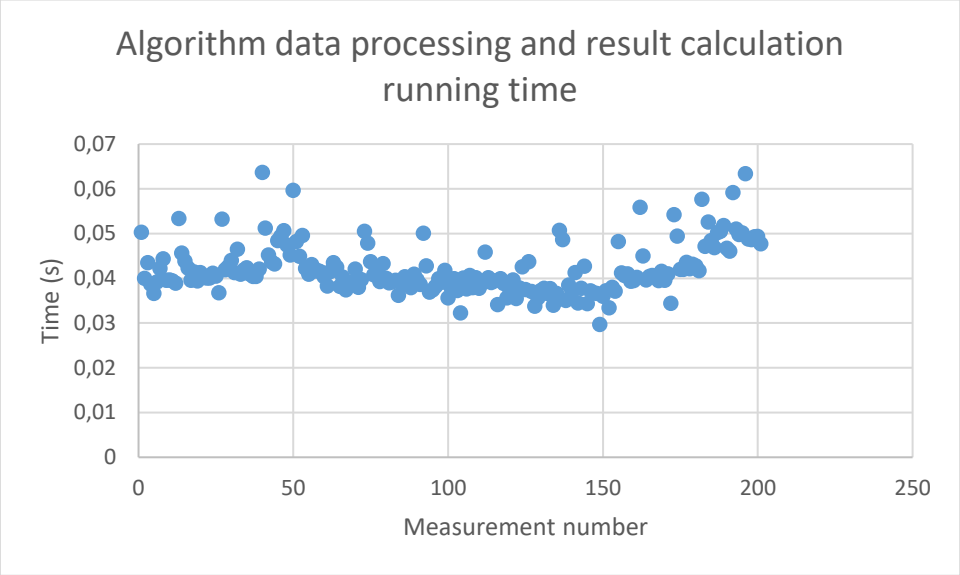


Figure 25. Time for data processing and result calculation. Mean time 0,042 s.

Moreover, Table 12 shows the time delays of different parts of the algorithm.

Table 12. Time delays of different parts of the algorithm.

Total cycle time (s)	Capture time window (s)	Capturing auxiliary time (s)	Data saving time (s)	Data manipulation time (s)
7,98	5	1,66	1,29	0,034
13,05	10	1,77	1,24	0,034
17,94	15	1,60	1,31	0,035

It can be seen that for a fixed capturing time window of 5, 10, and 15 s, the algorithm (data manipulation and data saving rows) plus the tshark functions (capturing auxiliary time) add approximately 3 s of delay, resulting in total cycles times of 7,98, 13,05, and 17,94 s. Depending on the final application requirements, this could be an acceptable overhead in terms of delay.

Figure 26 shows the memory usage results for different scenarios. The plot shows that the idle usage case has the lowest amount of memory usage, followed by the RPi working as an AP and with the highest memory usage when the RPi is working as an AP and also running the number of devices prediction script, which is according to the expectation. Furthermore, the periodical higher usage values occur when the Wireshark program is capturing the packets. As the memory usage parameter for the RPi command "top" command is updating every 1 s, this can explain the fluctuations that can be seen in the period of the memory usage for the use cases where the algorithm is running on top of the access point functionality.

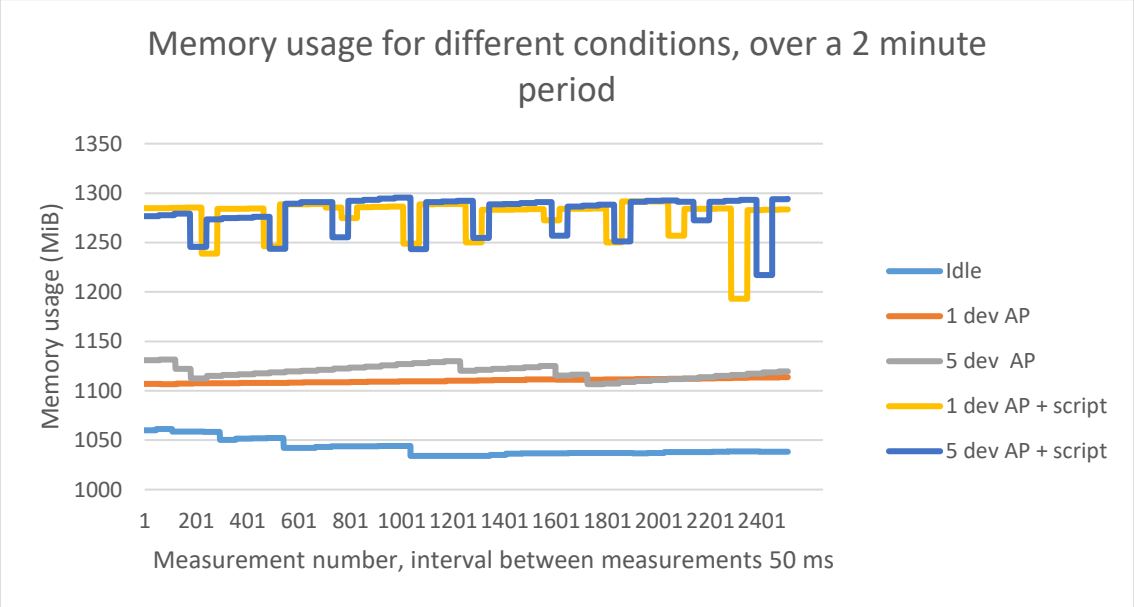


Figure 26. Memory usage of different scenarios.

In Table 13, the mean and maximum memory usage of different scenarios are shown.

Table 13. Mean, maximum and minimum memory usage of different scenarios.

	Memory usage (MiB)		
Scenario	Mean	Max	Min
Idle	1042,1	1061,3	1034,0
1 dev AP	1110,2	1113,8	1106,6
5 dev AP	1119,2	1131,5	1106,7
1 dev AP + algorithm	1277,4	1291,5	1193,2
5 dev AP + algorithm	1279,3	1295,5	1217,1

It can be seen that the memory usage when five client devices are connected to the RPi AP is on average 1,9 to 9,0 MiB higher than when only one device is connected, which is consistent with the expected result. Also, it can be seen that the algorithm adds on average 160,1 to 167,2 MiB of additional memory usage to the device.

5.4 Summary of the results

For the part of unsupervised learning (PCA and k-means), despite efforts with multiple combinations of observed input parameters, no currently usable representative relationship between the initially used parameters could be determined, nor any ways to reduce the dimensionality of the dataset where those parameters were used could be found. This means that the combination of the parameters monitored on a resource constrained device may benefit from further refinement to enhance their usability in the algorithm development, given the observed scenarios.

On the other hand, for the supervised learning method (linear regression), the main contributor to determining how many devices were connected to the network was the count of data traces obtained during a time period. This corresponds to the expectation, because each device's network traffic is added on top of the rest of the traffic and the more devices connected to the network, the more packets will be transferred. There were not many lost packets detected from the network traces, which can be attributed to the TCP's error handling capabilities. The average and median values of RTT did not play a significant role in the effect to the number of devices connected to the network, which might be due to the network not being constrained enough for these values to change. The algorithm running on a capture window of 5 s had a much lower accuracy value of 29,75%

compared to 10 s and 15 s capture windows (which had accuracy values of 94,67% and 93,62%, respectively), which is corresponding to the model estimations and the fact that the longer the collection and averaging time, the more stable the result will be.

Applying the same method but adding more connected client devices and applying more intense traffic to the network seemed to cause issues for the RPi, as it was not able to keep up with the traffic and it appeared to start to throttle from a certain amount of traffic; experiment done with higher speed devices, resulted in limits to output speed/packets transferred. Figure 27 illustrates this aspect, where it can be seen that with five devices connected, the count of measurements starts to decrease: When the median of the count of measurements increases from 1 to 4 devices (from 19536 to 29819), then for 5 devices it lower (29655), which is more closely corresponding to the range between 3 (32900) and 4 devices. This might be due to the limited computational power of the RPI4 device and additionally running both the AP and data monitoring tasks at the same time.

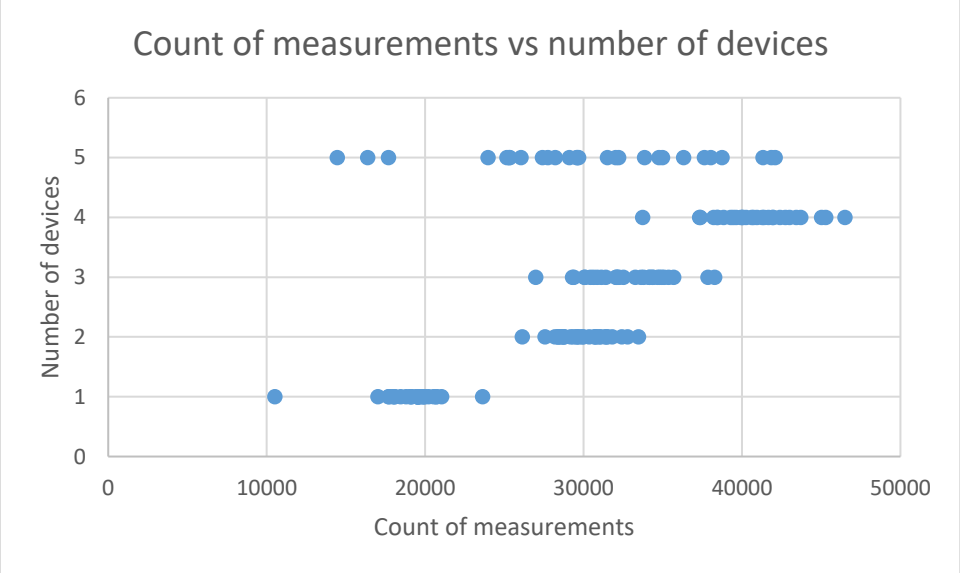


Figure 27. Count of measurements vs number of devices.

This chapter gave an overview of the results obtained while using different algorithms and the analysis of the results. The next chapter presents the details of the summary, lessons learned and future work.

6 Conclusion

6.1 Summary

This thesis addressed the issue of using a DDML approach as a first step towards performing network control and scheduling. The goal of the work was to develop a suitable algorithm, evaluate its performance, and deploy it on an RPi device. The minimal amount of data required to achieve (near) optimal algorithm performance was also evaluated based on the available data.

Given the research questions presented in the first chapter of this thesis, the answers found during the thesis are as follows: An algorithm can be proposed for network performance monitoring using supervised learning (e.g., linear regression); on the other hand, using unsupervised learning (e.g., PCA) proved to be inconclusive. The minimal amount of data required for (near) optimal operation of the algorithm was determined to be dependent on the dataset based on which the training would be performed. The proposed algorithm using linear regression can be deployed on an SBC and the impact on the memory usage and algorithm time consumption has been recorded.

The dataset for the analysis was obtained via real-life testing using a network setup with variable devices connected and different constraints applied to the network. The analysis and training the algorithm was done using Google Colaboratory and the implementation and testing (deployment) was done an a RPi 4.

The results showed that the unsupervised learning approach was not sufficiently compelling, as there were no clear cluster visible with k-means clustering and the PCA did not provide conclusive results with which to proceed with further analysis. To overcome this, several alternative methods were investigated, and linear regression was selected as a supervised learning approach. Linear regression using three parameters showed a mean accuracy of 94,67% for estimating the number of devices connected to the network using a 10 s capture time window. The average memory usage of the algorithm was measured as 160,1 MiB to 167,2 MiB and the average time usage of the

algorithm, when using a 10 s capture time window, was measured as 13,05 s, from which the algorithm output calculation time was 0,034 s.

While the obtained results illustrate the feasibility of the proposed approach, some limitations have also been identified. To improve and expand on the results, the next sections briefly summarize the lessons learned and provide future work prospects.

6.2 Lessons learned

Working on data analysis with moderately sized datasets and a moderate number of datasets, it is preferable to develop an automated solution for file analysis and visualization, as would be the case with large datasets, as this can save a significant amount of time. General results show that even if selecting the hardware is done based on some research, there may still appear some unexpected challenges that are not mentioned in the scientific literature, for which creative solutions are needed. Considering state of the art, some problems require creating new datasets and creating or adjusting algorithms to suit the specific problem, as often the information that is available is quite general and difficult to apply on a specific problem.

Taking into account both the results and lessons learned, the future work perspectives are provided in the next section.

6.3 Future work

To expand on and improve the problem solving considered in this thesis, the first option is to use a compatible Wi-Fi adapter that supports monitoring, access point modes and a larger amount of network client device connections (at least 25) in access point mode. This solution would give the opportunity to first test a larger amount of device connections in access point mode, to verify if the algorithm will indeed work with more than five connected client devices. Additionally, using the AP device with an external adapter only in monitoring mode might reduce memory usage and possibly current consumption, as then the device does not have to additionally act as a hotspot. One of the challenges with using monitoring mode will be to handle encrypted data, for which there is a possibility in Wireshark to include the encryption keys extracted from a known

network’s devices, but this remains to be investigated if and what impact this might have on the algorithm’s performance.

Further perspectives could be to adapt the algorithm to additional SBC devices, to assess how transferrable the algorithm is and if there are other devices that might provide improved performance.

Also, it could be considered to use different devices with different traffic patterns connected to the network and determine how transferrable the algorithm will be for those situations. This will additionally allow to acquire more datasets with increase variation, to better adjust the algorithm.

Using devices or adapters with extended possibilities could provide an opportunity to explore the usability of alternative methods and algorithms, to potentially improve the solution.

Another further future work opportunity might be to use alternative libraries and frameworks for the training and deployment of the ML models, to further optimize the algorithm.

Energy consumption impact of the developed solution and possible alternative solutions could be explored in future work as well as optimizing the existing algorithm to work with a shorter capturing time window to enhance the reaction time of the algorithm.

Finally, the algorithm could be upgraded by improving and adding the network controlling and scheduling functionalities.

References

[1] S. Jaffry, S. T. Shah, and S. F. Hasan, “Data-Driven Semi-Supervised Anomaly Detection Using Real-World Call Data Record,” in *2020 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2020 - Proceedings*, 2020. doi: 10.1109/WCNCW48565.2020.9124782.

[2] T. Wang, S. Wang, and Z. H. Zhou, “Machine learning for 5G and beyond: From model-based to data-driven mobile wireless networks,” *China Communications*, vol. 16, no. 1, 2019.

[3] Z. Hou, H. Gao, and F. L. Lewis, “Data-driven control and learning systems,” *IEEE Transactions on Industrial Electronics*, vol. 64, no. 5, 2017, doi: 10.1109/TIE.2017.2653767.

[4] G. Baggio, D. S. Bassett, and F. Pasqualetti, “Data-driven control of complex networks,” *Nat Commun*, vol. 12, no. 1, 2021, doi: 10.1038/s41467-021-21554-0.

[5] A. Salh *et al.*, “A Survey on Deep Learning for Ultra-Reliable and Low-Latency Communications Challenges on 6G Wireless Systems,” *IEEE Access*, vol. 9. 2021. doi: 10.1109/ACCESS.2021.3069707.

[6] S. Szott *et al.*, “Wi-Fi Meets ML: A Survey on Improving IEEE 802.11 Performance with Machine Learning,” *IEEE Communications Surveys and Tutorials*, vol. 24, no. 3, 2022, doi: 10.1109/COMST.2022.3179242.

[7] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, “A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4. 2017. doi: 10.1109/COMST.2017.2727878.

[8] F. Azmat, Y. Chen, and N. Stocks, “Predictive modelling of RF energy for wireless powered communications,” *IEEE Communications Letters*, vol. 20, no. 1, 2016, doi: 10.1109/LCOMM.2015.2497306.

[9] A. Martin *et al.*, “Network resource allocation system for QoE-aware delivery of media services in 5G networks,” *IEEE Transactions on Broadcasting*, vol. 64, no. 2, 2018, doi: 10.1109/TBC.2018.2828608.

[10] J. Liu, R. Deng, S. Zhou, and Z. Niu, “Seeing the unobservable: Channel learning for wireless communication networks,” in *2015 IEEE Global Communications Conference, GLOBECOM 2015*, 2015. doi: 10.1109/GLOCOM.2014.7417805.

[11] S. J. Nawaz, S. K. Sharma, S. Wyne, M. N. Patwary, and M. Asaduzzaman, “Quantum Machine Learning for 6G Communication Networks: State-of-the-Art and Vision for the Future,” *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2909490.

[12] C. Sun and C. Yang, “Learning to Optimize with Unsupervised Learning: Training Deep Neural Networks for URLLC,” in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, 2019. doi: 10.1109/PIMRC.2019.8904143.

[13] M. S. Parwez, D. B. Rawat, and M. Garuba, “Big data analytics for user-activity analysis and user-anomaly detection in mobile wireless network,” *IEEE Trans Industr Inform*, vol. 13, no. 4, pp. 2058–2065, Aug. 2017, doi: 10.1109/TII.2017.2650206.

[14] Z. Liao, R. Zhang, S. He, D. Zeng, J. Wang, and H. J. Kim, “Deep Learning-Based Data Storage for Low Latency in Data Center Networks,” *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2901742.

[15] E. Balevi and R. D. Gitlin, “Unsupervised machine learning in 5G networks for low latency communications,” in *2017 IEEE 36th International Performance Computing and Communications Conference, IPCCC 2017*, 2018. doi: 10.1109/PCCC.2017.8280492.

[16] J. Gazda, E. Slapak, G. Bugar, D. Horvath, T. Maksymyuk, and M. Jo, “Unsupervised Learning Algorithm for Intelligent Coverage Planning and Performance Optimization of Multitier Heterogeneous Network,” *IEEE Access*, vol. 6, 2018, doi: 10.1109/ACCESS.2018.2847609.

[17] A. T. Z. Kasgari and W. Saad, “Model-Free Ultra Reliable Low Latency Communication (URLLC): A Deep Reinforcement Learning Framework,” in *IEEE International Conference on Communications*, 2019. doi: 10.1109/ICC.2019.8761721.

[18] Y. Ling, B. Yi, and Q. Zhu, “An Improved Vertical Handoff Decision Algorithm for Heterogeneous Wireless Networks.”

[19] K. Hamidouche, A. T. Z. Kasgari, W. Saad, M. Bennis, and M. Debbah, “Collaborative artificial intelligence (AI) for user-cell association in ultra-dense cellular systems,” in *2018 IEEE International Conference on Communications Workshops, ICC Workshops 2018 - Proceedings*, 2018. doi: 10.1109/ICCW.2018.8403664.

[20] G. Han, L. Xiao, and H. V. Poor, “Two-dimensional anti-jamming communication based on deep reinforcement learning,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017. doi: 10.1109/ICASSP.2017.7952524.

[21] L. T. Tan and R. Q. Hu, “Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning,” *IEEE Trans Veh Technol*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018, doi: 10.1109/TVT.2018.2867191.

[22] H. Yang, J. Zhao, K. Y. Lam, Z. Xiong, Q. Wu, and L. Xiao, “Distributed Deep Reinforcement Learning Based Spectrum and Power Allocation for Heterogeneous Networks,” *IEEE Trans Wirel Commun*, 2022, doi: 10.1109/TWC.2022.3153175.

[23] X. Kong *et al.*, “Deep Reinforcement Learning-Based Energy-Efficient Edge Computing for Internet of Vehicles,” *IEEE Trans Industr Inform*, vol. 18, no. 9, 2022, doi: 10.1109/TII.2022.3155162.

[24] B. Ma, W. Guo, and J. Zhang, “A Survey of Online Data-Driven Proactive 5G Network Optimisation Using Machine Learning,” *IEEE Access*, vol. 8. 2020. doi: 10.1109/ACCESS.2020.2975004.

[25] S. Narejo and E. Pasero, “An application of internet traffic prediction with deep neural network,” in *Smart Innovation, Systems and Technologies*, vol. 69, 2017. doi: 10.1007/978-3-319-56904-8_14.

[26] Y. Hua, Z. Zhao, Z. Liu, X. Chen, R. Li, and H. Zhang, “Traffic Prediction Based on Random Connectivity in Deep Learning with Long Short-Term Memory,” in *IEEE Vehicular Technology Conference*, 2018. doi: 10.1109/VTCFall.2018.8690851.

[27] D. Tikunov and T. Nishimura, “Traffic prediction for mobile network using Holt-Winter’s exponential smoothing,” in *2007 15th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2007*, 2007. doi: 10.1109/SOFTCOM.2007.4446113.

[28] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das, “Understanding traffic dynamics in cellular data networks,” in *Proceedings - IEEE INFOCOM*, 2011. doi: 10.1109/INFCOM.2011.5935313.

[29] F. Xu *et al.*, “Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach,” *IEEE Trans Serv Comput*, vol. 9, no. 5, 2016, doi: 10.1109/TSC.2016.2599878.

[30] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, “The Learning and Prediction of Application-Level Traffic Data in Cellular Networks,” *IEEE Trans Wirel Commun*, vol. 16, no. 6, 2017, doi: 10.1109/TWC.2017.2689772.

[31] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui, “Spatio-Temporal Wireless Traffic Prediction with Recurrent Neural Network,” *IEEE Wireless Communications Letters*, vol. 7, no. 4, 2018, doi: 10.1109/LWC.2018.2795605.

[32] L. V. Le, D. Sinh, L. P. Tung, and B. S. P. Lin, “A practical model for traffic forecasting based on big data, machine-learning, and network KPIs,” in *CCNC 2018 - 2018 15th IEEE Annual Consumer Communications and Networking Conference*, 2018. doi: 10.1109/CCNC.2018.8319255.

[33] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, “Characterizing and modeling internet traffic dynamics of cellular devices,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 39, no. 1, 2011, doi: 10.1145/2007116.2007148.

[34] S. Zhang *et al.*, “Traffic Prediction Based Power Saving in Cellular Networks: A Machine Learning Method,” in *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, 2017. doi: 10.1145/3139958.3140053.

[35] B. Yang, W. Guo, B. Chen, G. Yang, and J. Zhang, “Estimating Mobile Traffic Demand Using Twitter,” *IEEE Wireless Communications Letters*, vol. 5, no. 4, 2016, doi: 10.1109/LWC.2016.2561924.

[36] S. S. Sepasgozar and S. Pierre, “Network Traffic Prediction Model Considering Road Traffic Parameters Using Artificial Intelligence Methods in VANET,” *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3144112.

[37] F. Botta, H. S. Moat, and T. Preis, “Quantifying crowd size with mobile phone and Twitter data,” *R Soc Open Sci*, vol. 2, no. 5, 2015, doi: 10.1098/rsos.150162.

[38] A. Knapińska, P. Lechowicz, and K. Walkowiak, “Machine-Learning Based Prediction of Multiple Types of Network Traffic,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021. doi: 10.1007/978-3-030-77961-0_12.

[39] M. S. Parwez, D. B. Rawat, and M. Garuba, “Big data analytics for user-activity analysis and user-anomaly detection in mobile wireless network,” *IEEE Trans Industr Inform*, vol. 13, no. 4, pp. 2058–2065, Aug. 2017, doi: 10.1109/TII.2017.2650206.

[40] L. V. Le, D. Sinh, B. S. P. Lin, and L. P. Tung, “Applying Big Data, Machine Learning, and SDN/NFV to 5G Traffic Clustering, Forecasting, and Management,” in *2018 4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018*, 2018. doi: 10.1109/NETSOFT.2018.8460129.

[41] M. Xu, Q. Wang, and Q. Lin, “Hybrid holiday traffic predictions in cellular networks,” in *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, 2018. doi: 10.1109/NOMS.2018.8406291.

[42] T. Zhang, K. Zhu, and E. Hossain, “Data-Driven Machine Learning Techniques for Self-Healing in Cellular Wireless Networks: Challenges and Solutions,” *Intelligent Computing*, vol. 2022, 2022, doi: 10.34133/2022/9758169.

[43] C. Sun and C. Yang, “Unsupervised deep learning for ultra-reliable and low-latency communications,” in *2019 IEEE Global Communications Conference, GLOBECOM 2019 - Proceedings*, 2019. doi: 10.1109/GLOBECOM38437.2019.9013851.

[44] M. Ferriol-Galmés *et al.*, “Building a Digital Twin for network optimization using Graph Neural Networks,” *Computer Networks*, vol. 217, 2022, doi: 10.1016/j.comnet.2022.109329.

[45] S. Farthofer, M. Herlich, C. Maier, S. Pochaba, J. Lackner, and P. Dorfinger, “An Open Mobile Communications Drive Test Data Set and Its Use for Machine Learning,” *IEEE Open Journal of the Communications Society*, vol. 3, 2022, doi: 10.1109/OJCOMS.2022.3210289.

[46] A. Delimargas *et al.*, “Evaluating a modified PCA approach on network anomaly detection,” in *International Conference on Next Generation Networks and Services, NGNS*, 2014. doi: 10.1109/NGNS.2014.6990240.

[47] A. U. Chaudhry, “Using machine learning to find the hidden relationship between RTT and TCP throughput in WiFi,” *EURASIP J Wirel Commun Netw*, vol. 2021, no. 1, 2021, doi: 10.1186/s13638-021-02076-1.

[48] M. Arlitt, B. Krishnamurthy, and J. C. Mogul, “Predicting short-transfer latency from TCP arcana: A trace-based validation,” in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, 2005.

[49] G. Hu, T. Zhou, and Q. Liu, “Data-Driven Machine Learning for Fault Detection and Diagnosis in Nuclear Power Plants: A Review,” *Front Energy Res*, vol. 9, 2021, doi: 10.3389/fenrg.2021.663296.

[50] Smola A and Vishwanathan S.V.N, *Introduction to Machine Learning*. 2008.

[51] G. S. Andreas C. Muller, *Introduction to Machine Learning with Python: a guide for data scientist*. 2017.

[52] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, “OPTICS: Ordering Points to Identify the Clustering Structure,” *SIGMOD Record (ACM Special Interest Group on Management of Data)*, vol. 28, no. 2, 1999, doi: 10.1145/304181.304187.

[53] “Lesson 4: Multivariate Normal Distribution.” Accessed: Dec. 29, 2023. [Online]. Available: https://online.stat.psu.edu/stat505/book/export/html/636

[54] F. T. Liu, K. M. Ting, and Z. H. Zhou, “Isolation forest,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2008. doi: 10.1109/ICDM.2008.17.

[55] O. Alghushairy, R. Alsini, T. Soule, and X. Ma, “A review of local outlier factor algorithms for outlier detection in big data streams,” *Big Data and Cognitive Computing*, vol. 5, no. 1. 2021. doi: 10.3390/bdcc5010001.

[56] R. Bro and A. K. Smilde, “Principal component analysis,” *Analytical Methods*, vol. 6, no. 9. 2014. doi: 10.1039/c3ay41907j.

[57] N. N. Srinidhi, S. M. Dilip Kumar, and K. R. Venugopal, “Network optimizations in the Internet of Things: A review,” *Engineering Science and Technology, an International Journal*, vol. 22, no. 1. 2019. doi: 10.1016/j.jestch.2018.09.003.

[58] D. E. Comer and D. L. Stevens, “Internetworking with TCP/IP, Volume III: Client-Server Programming and Applications, Second Edition,” *IEEE Network*, vol. 10, no. 4. 1996. doi: 10.1109/MNET.1996.527010.

[59] L. L. Peterson and B. S. Davie, “Computer Networks: A System Approach,” *IEEE Communications Magazine*, vol. 36, no. 5, 2005, doi: 10.1109/mcom.1998.667947.

[60] “pandas.DataFrame — pandas 2.1.4 documentation.” Accessed: Dec. 29, 2023. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

[61] ITU-T, “Network performance objectives for IP-based services,” 2011.

[62] “Raspberry Pi 4 Model B – Raspberry Pi.” Accessed: Dec. 20, 2023. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[63] Nvidia, “DATA SHEET NVIDIA Jetson Nano System-on-Module Maxwell GPU + ARM Cortex-A57 + 4GB LPDDR4 + 16GB eMMC,” 2014, Accessed: Dec. 20, 2023. [Online]. Available: www.khronos.org/conformance.

[64] “Dev Board | Coral.” Accessed: Dec. 20, 2023. [Online]. Available: https://coral.ai/products/dev-board/

[65] “Raspberry Pi Zero W – Raspberry Pi.” Accessed: Dec. 20, 2023. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-zero-w/

[66] “Raspberry Pi 4 Model B/4GB - PiShop.ca.” Accessed: Dec. 20, 2023. [Online]. Available: https://www.pishop.ca/product/raspberry-pi-4-model-b-4gb/

[67] “Raspberry Pi Pico – Raspberry Pi.” Accessed: Dec. 20, 2023. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-pico/

[68] “Raspberry Pi Pico W – Pi Australia.” Accessed: Dec. 20, 2023. [Online]. Available: https://raspberry.piaustralia.com.au/products/raspberry-pi-pico-w

[69] P. Warden, *TinyML : machine learning with Tensorflow Lite on Arduino, and ultra-low power micro-controllers / Pete Warden and Daniel Situnayake*. 2020.

Appendix 1 – PCA additional results

In Figure 28 is the PCA scree plot for 9 parameters used. The parameters used can be seen in .

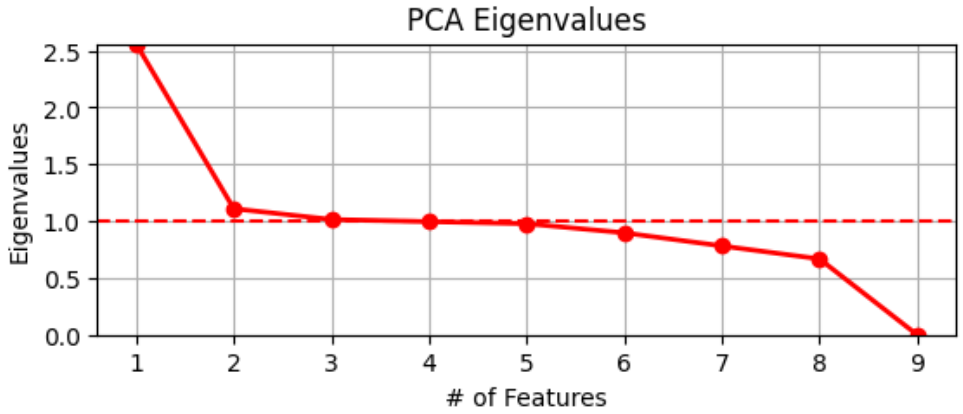


Figure 28. PCA Scree plot for 9 parameters.

In Figure 29 is the 2 component PCA plot for 9 parameters used.

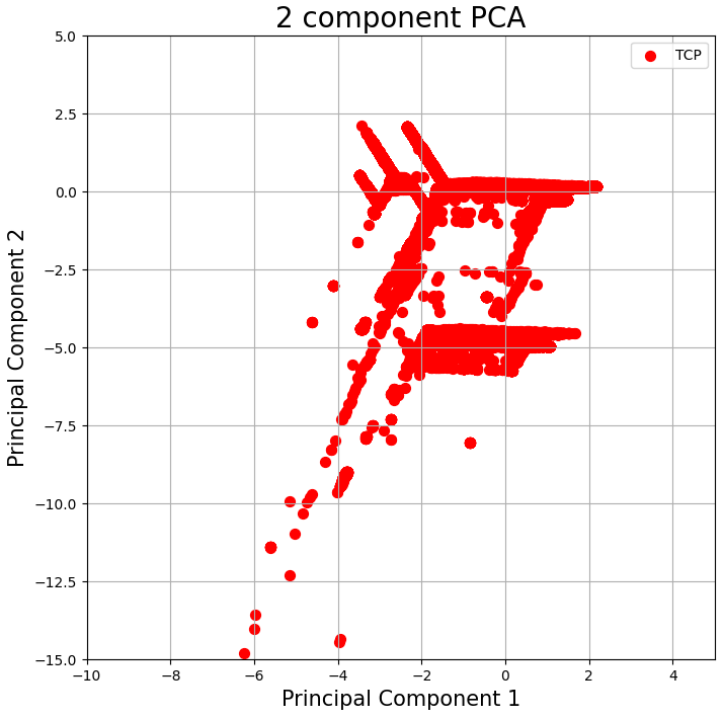


Figure 29. 2 component PCA for 9 parameters.

In Table 14 is the summary of variance ratios for the principal components separately and the explained variance for each parameter.

Table 14. PCA results for 9 parameters.

	Var ratio	Delay	Length	Seq	Ack	Win	Len	Ret	Dup	Lost
PC_1	0,28	0,04	0,59	0,37	0,26	0,24	0,59	0,05	0,18	0,01
PC_2	0,12	0,14	0,03	0,03	0,54	0,56	0,03	0,60	0,03	0,05
PC_3	0,11	0,69	0,01	0,04	0,11	0,15	0,01	0,01	0,33	0,61
PC_4	0,11	0,33	0,02	0,02	0,24	0,14	0,02	0,31	0,46	0,71

In Figure 30 is the PCA scree plot for a dataset with added 500 ms delay for 50% of the measurement time.

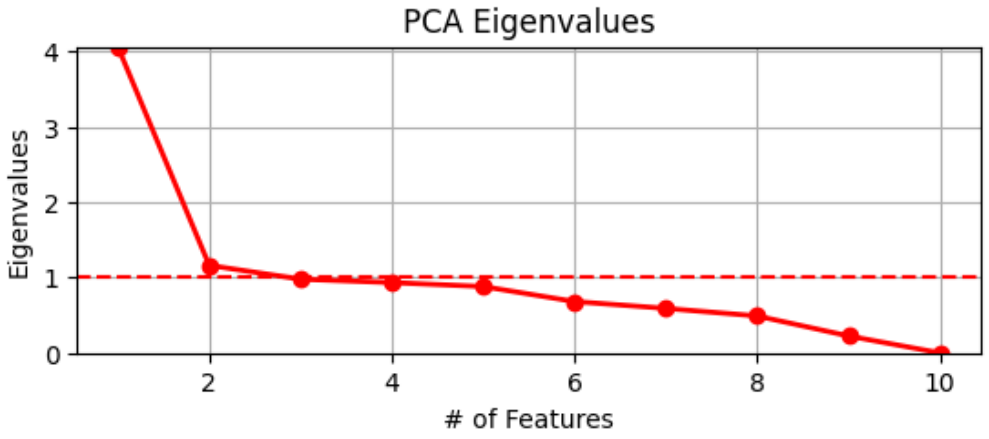


Figure 30. PCA Scree plot for dataset with added 500 ms delay for 50% of the time.

In Figure 31 is the 2 component PCA for a dataset with added 500 ms delay for 50% of the measurement time.

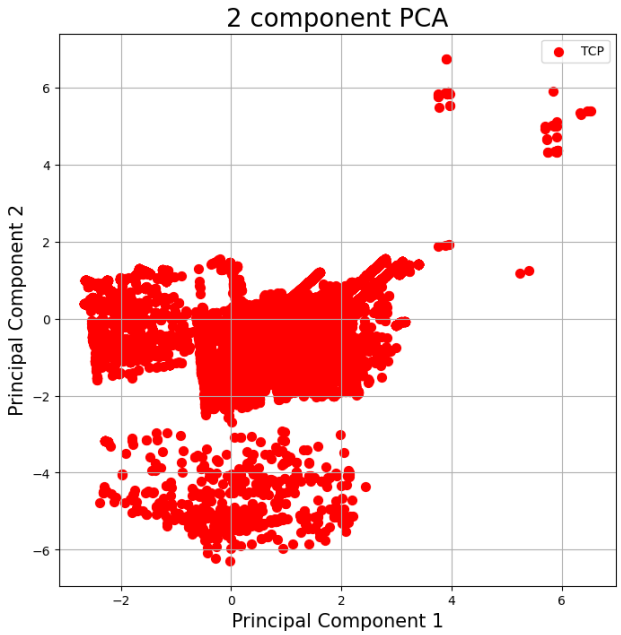


Figure 31. 2 component PCA for dataset with added 500 ms delay for 50% of the time.

In Table 15 is the summary of variance ratios for the principal components separately and the explained variance for each parameter for a dataset with added 500 ms delay for 50% of the measurement time.

Table 15. PCA variance results for datset with added 500 ms delay for 50% of the time.

	Var ratio	Time	Length	Seq	Ack	Win	Len	TSval	TSecr	Ret	Dup
PC_1	0,406	0,042	0,448	0,299	0,278	0,271	0,449	0,409	0,409	0,001	0,144
PC_2	0,116	0,587	0,196	0,185	0,361	0,331	0,189	0,109	0,017	0,537	0,080
PC_3	0,098	0,517	0,048	0,134	0,105	0,164	0,047	0,066	0,023	0,668	0,471
PC_4	0,093	0,430	0,011	0,151	0,147	0,209	0,014	0,077	0,061	0,283	0,798

Appendix 2 – Linear Regression python algorithm source code

```
import time, os, sys
import pandas as pd
from datetime import datetime

#parameters 10s
a1 = 0.0028318019
a2 = 90.3946069
a3 = -0.000750874772
a4 = -0.4
F = 0.0

#tshark commands for pcap capture and csv save
command1 = "sudo tshark -w /tmp/test1.pcap -a duration:10 -i wlan0 -f
tcp -T fields -e frame.number -e ip.proto -e ftp -e ftp-data -e
tcp.analysis.ack_rtt -e tcp.analysis.retransmission -E header=y -E
separator=, -E quote=d -E occurrence=f"
command2 = "sudo tshark -r /tmp/test1.pcap -i wlan0 -f tcp -T fields -
e frame.number -e ip.proto -e ftp -e ftp-data -e tcp.analysis.ack_rtt
-e tcp.analysis.retransmission -Y ip.proto==6 -E header=y -E
separator=, -E quote=d -E occurrence=f > /tmp/test1.csv"

while True:
    #Initial timestamp
    time3 = time.time()
    os.system("/bin/bash -c \"" + command1 + "\"")
    #Timestamp after tshark capturing
    time4 = time.time()
    os.system("/bin/bash -c \"" + command2 + "\"")

    #Timestamp after tshark save to csv
    time1 = time.time()
    #Read csv to dataframe
    df = pd.read_csv('/tmp/test1.csv', encoding = "ISO-8859-1")

    #Dataframe filter out required data
    df = df[df['ip.proto'] == 6]
    df = df[df['ftp'] != 'ftp']
    df = df[df['ftp-data'].isna()]

    #add "0" and "1"
    df['tcp.analysis.retransmission'] =
df['tcp.analysis.retransmission'].notnull().astype('int')
```

```
    #Count rows, calculate median of RTT, sum of retransmission
packets
    Count = len(df)
    Median = df['tcp.analysis.ack_rtt'].median()
    Retransmission_count = df['tcp.analysis.retransmission'].sum()

    #Calculate number of devices
    F = a1 * Count + a2 * Median + a3 * Retransmission_count + a4
    #Timestamp after calculations
    time2 = time.time()

    #Count different delays
    totaltime = time2 - time1
    totaltime2 = time2 - time3
    totaltime3 = time2 - time4
    #Round final value
    roundF = round(F)
    print('Count of rows', Count, 'Median', Median, 'Count of
retransmission packets', Retransmission_count)

    #Current date and time acquiring and printing
    dt = datetime.now()
    str_dt = dt.strftime("%d-%m-%Y, %H:%M:%S")
    str_dt2 = dt.strftime("%d-%m-%Y,%H:%M:%S")
    strstats = str_dt2 + ",Count," + str(Count) + ",Median," +
str(Median) + ",Retransmission," + str(Retransmission_count) +
",No_of_devices," + str(roundF) + ",time_delay," + str(totaltime) +
",total_time_delay," + str(totaltime2) +
",total_time_delay_minus_capture," + str(totaltime3) + "\n"

    L = "Number of connected devices - " + str(F) +"\n"
    s = "   "
    #Print number of devices
    print(dt, ' - ', L)
    tt = " - time - " + str(totaltime)

    #Save to log
    file1 = open("/home/john/logfile.txt", "a")
    file1.writelines(str_dt)
    file1.writelines(s)
    file1.writelines(tt)
    file1.writelines(s)
    file1.writelines(L)
    file1.close
    file2 = open("/home/john/stats.txt", "a")
    file2.writelines(strstats)
    file2.close
```

Figure 32. RPi algorithm source code

Appendix 3 – Raspberry Pi Pico W MicroPython source code

```
import network
import socket
from time import sleep
from picozero import pico_led
import machine
from ftplib import FTP
#Raspberry Pi Foundation tutorial code used as a basis

ssid = "RPi_test"
password = "Password"

#Connect to Wi-Fi hotspot
def connect():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while wlan.isconnected() == False:
        print('Waiting for connection...')
        sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip

#Open IP socket
def open_socket(ip):
    address = (ip, 80)
    connection = socket.socket()
    connection.bind(address)
    connection.listen(1)
    return(connection)

#Periodically download 80KB file
def serve (connection):
    pico_led.off()
    while True:
        file=open("data2.csv","wb")
        print('Testing')
        pico_led.on()
        ftp = FTP('ftp.rebex.net', 21, 'demo', 'password')
        ftp.retrbinary('RETR ' +'/pub/example/WinFormClient.png',
file.write)
        file.close()
        ftp.quit()
        pico_led.off()
```

```
        sleep(0.5)

#Run defined functions
try:
    ip = connect()
    connection = open_socket(ip)
    serve(connection)
except KeyboardInterrupt:
    machine.reset()
```

Figure 33. RPi Pico W MicroPython source code.

Appendix 4 – Linear Regression PCA analysis

In Figure 34 is the resulting scatter plot for 4 principal components for a combined dataset. In Figure 35 is a 2 component PCA plot for a final set of parameters for a capture time window of 10 s, with added device number indicators.

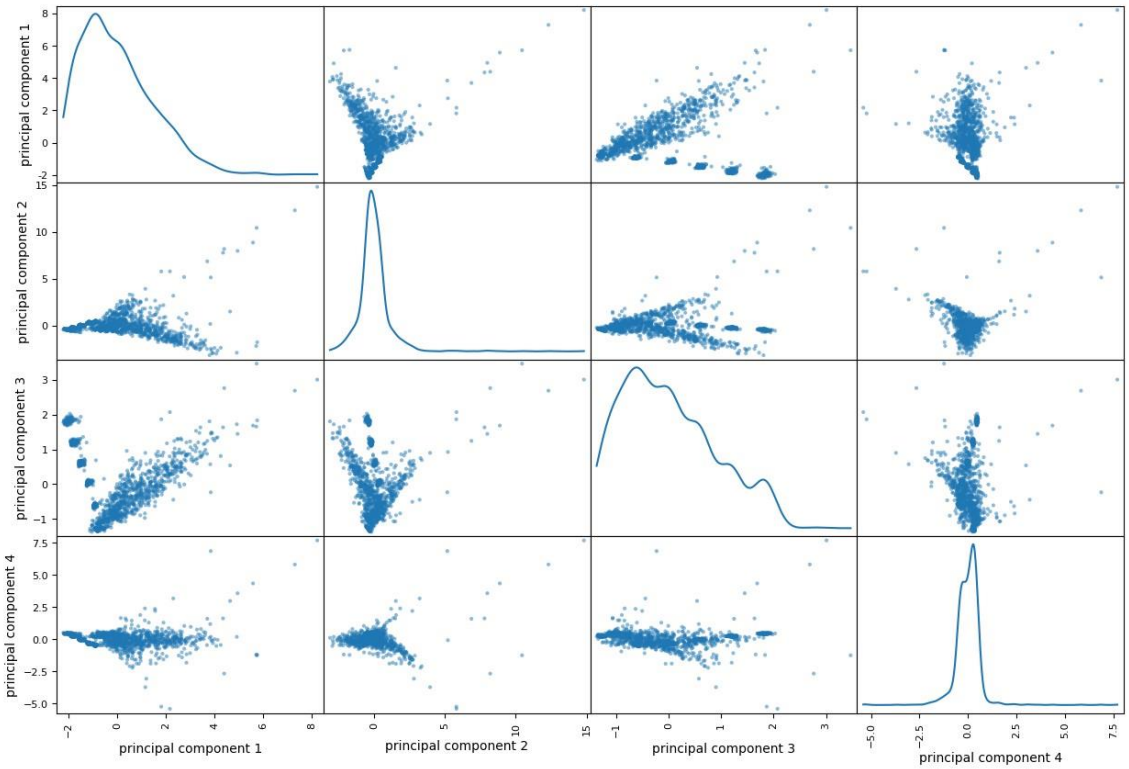


Figure 34. Scatter matrix for 4 component PCA for final set of parameters with time window of 10 seconds for a combined dataset.

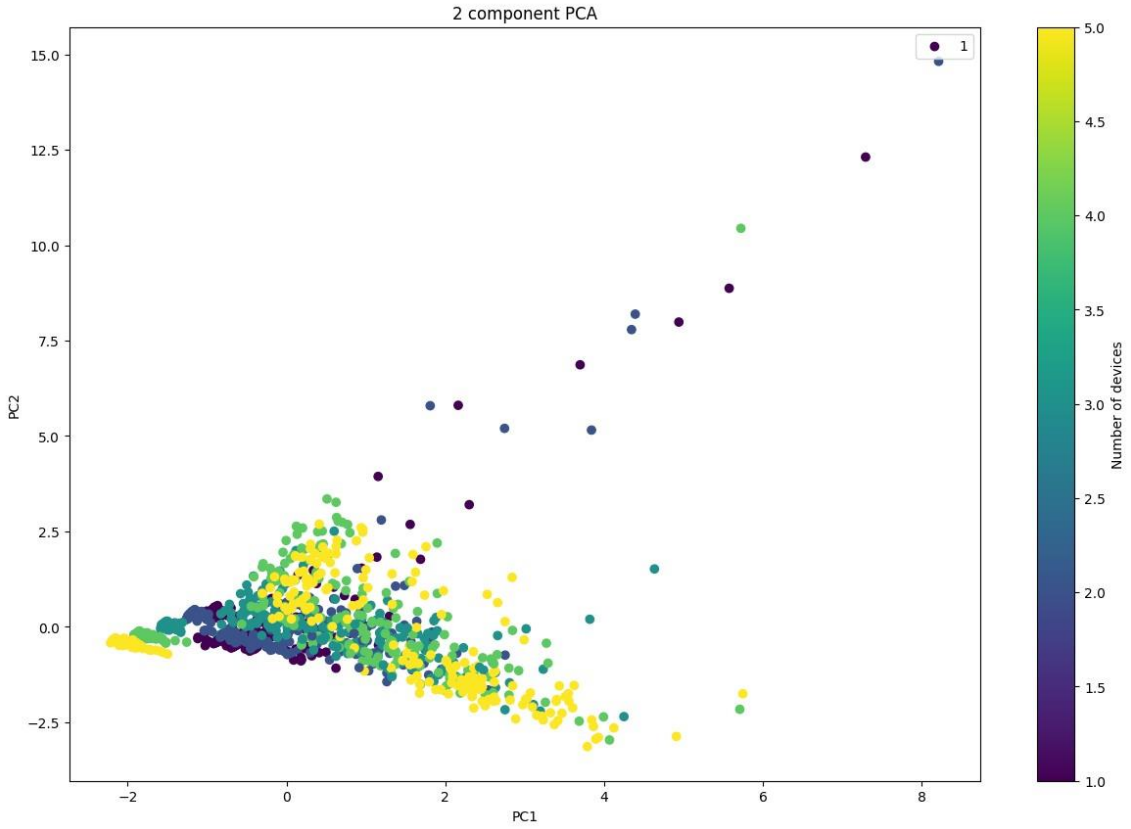


Figure 35. 2 component PCA plot for final set of parameters with time window of 10 seconds for a combined dataset with added device number indicators.

Table 16 displays the summary of variance ratios for the principal components separately and the explained variance for each parameter.

Table 16. 4 component PCA details for final set of parameters with time window of 10 seconds for a combined dataset.

	variance_ratio	Count	Average	Median	dup_count	retrans_count
PC_1	0,444341874	0,381042032	0,366792641	0,242999846	0,584044692	0,565785
PC_2	0,290423077	0,152872313	0,559752201	0,632071941	0,351243817	0,374727
PC_3	0,15987833	0,882944079	0,072960659	0,370783296	0,146300821	0,23707
PC_4	0,089158006	0,210017713	0,735934698	0,632740844	0,108010733	0,047596

Appendix 5 – Linear Regression additional plots

1. For 5 s time window

Figure 36 shows the count of measured packets vs number of devices for a 5 s capture window and combined dataset.

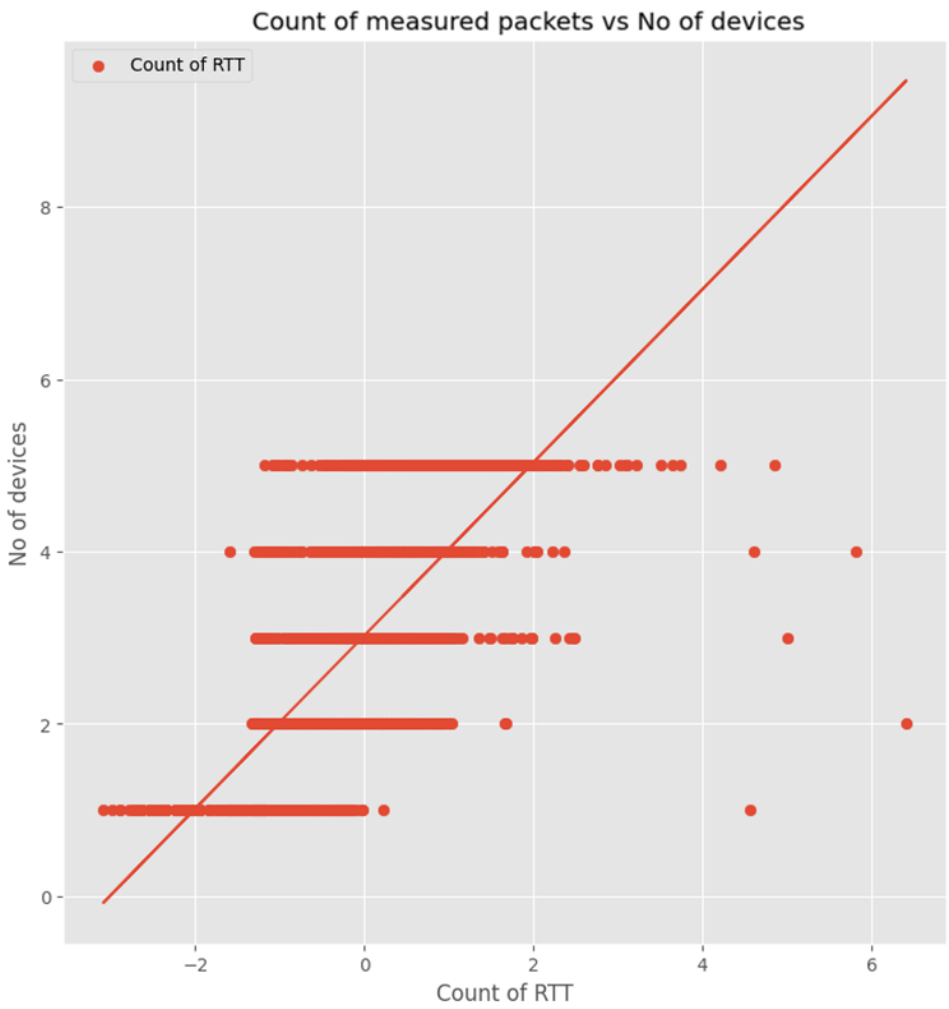


Figure 36. Count of measured packets vs number of devices for a 5 s capture window and combined dataset.

Figure 37 shows the count of suspected retransmission packets vs number of devices for a 5 s capture window and combined dataset.

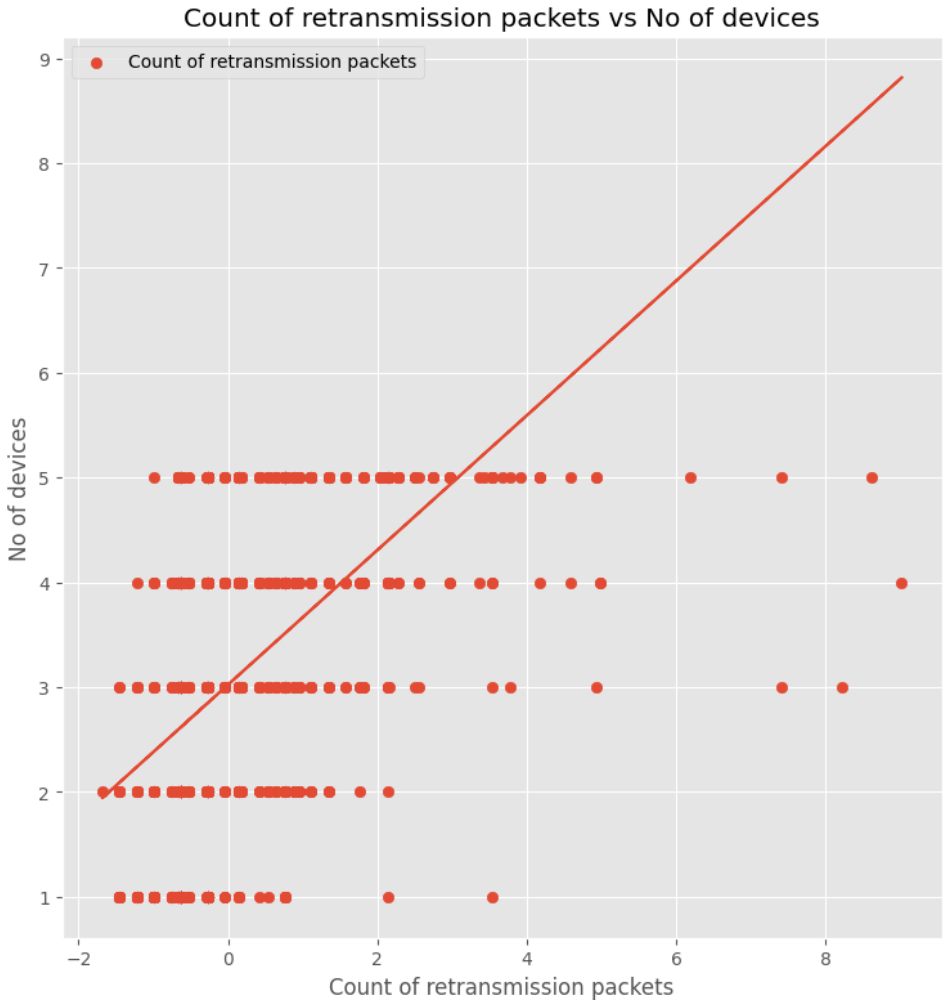


Figure 37. Count of retransmission packets vs number of devices for a 5 s capture window and combined dataset.

2. For 10 s time window

Figure 38 shows the count of duplicate ACKs vs number of devices for a 10 s capture window and combined dataset.

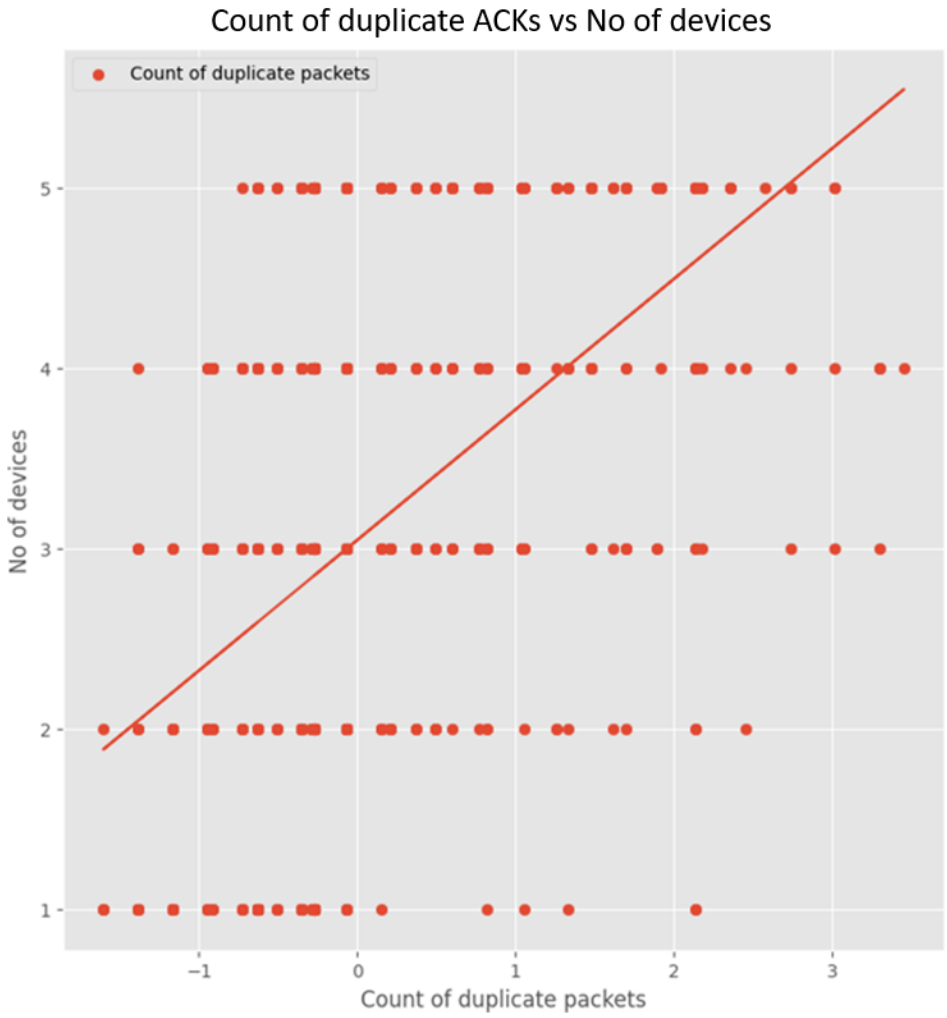


Figure 38. Count of duplicate ACKs vs number of devices for a 10 s time window and combined dataset.

Figure 39 shows the average of RTT vs number of devices for a 10 s capture window and combined dataset.

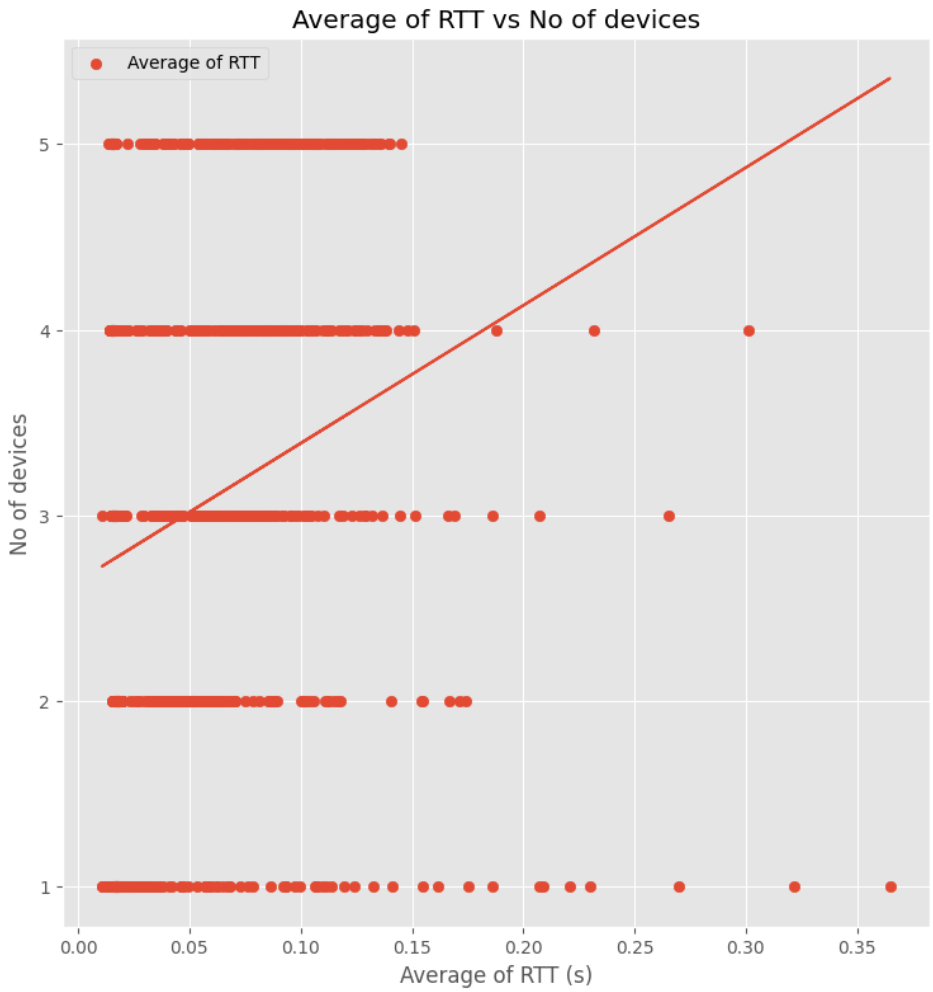


Figure 39. Average RTT vs number of devices for a 10 s time window and combined dataset.

3. For 15 s time window

Figure 40 shows the count of measured packets vs number of devices for a 15 s capture window and combined dataset.

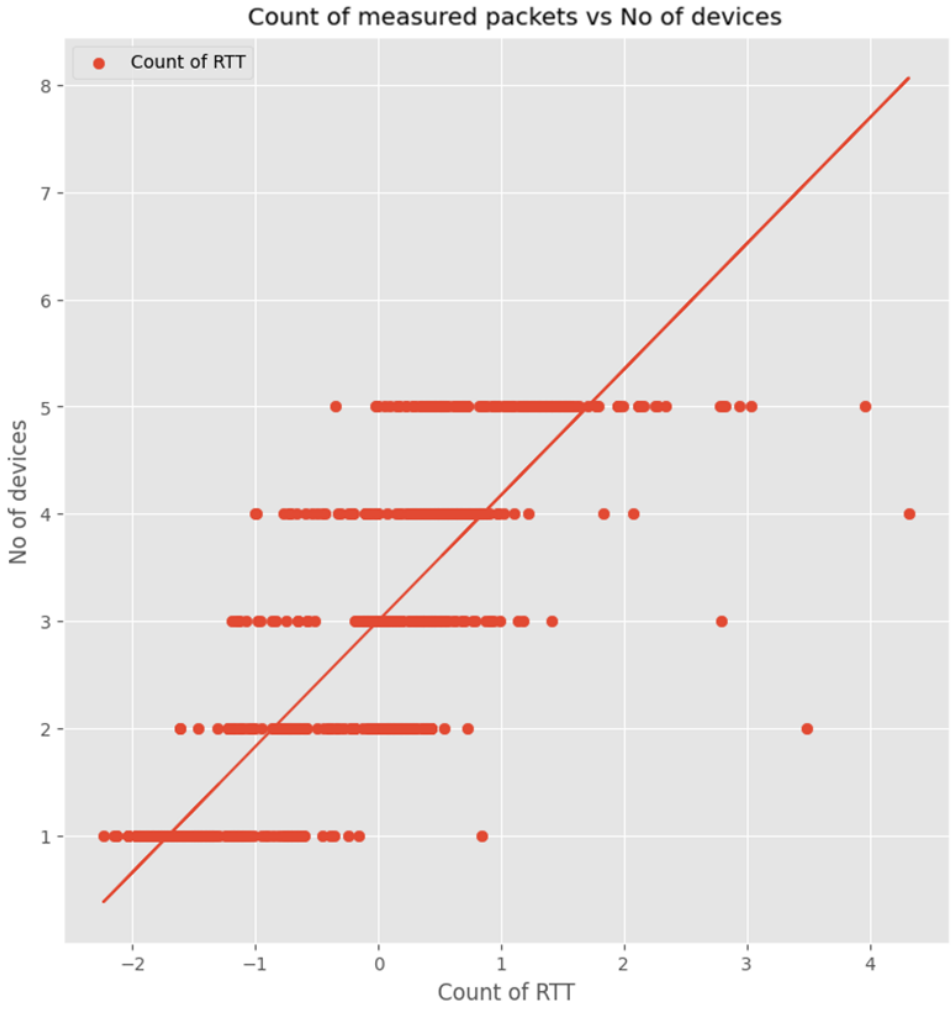


Figure 40. Count of measured packets vs number of devices for a 15 s capture window and combined dataset.

Figure 41 shows the count of suspected retransmission packets vs number of devices for a 15 s capture window and combined dataset.

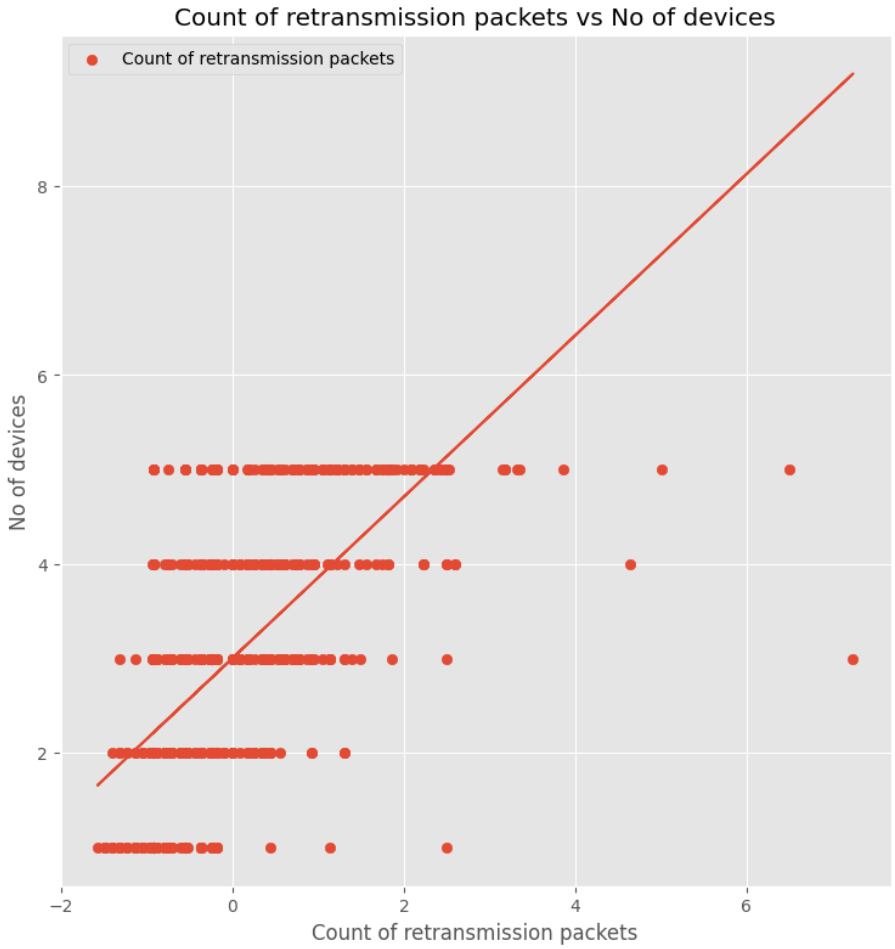


Figure 41. Count of retransmission packets vs number of devices for a 15 s capture window and combined dataset.

Appendix 6 – PCA source code

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

#Open the dataset and load it to the dataframe
url =
'https://raw.githubusercontent.com/kaarel2/test/main/RPi_Traces/Combin
ed3_retransmission.csv'
df = pd.read_csv(url, names=['Delay', 'Length', 'Seq', 'Ack', 'Win',
'Len', 'Retransmission', 'Duplicate', 'Lost_packets',
'cat_technology'])

#Define features
features = ['Delay', 'Length', 'Seq', 'Ack', 'Win', 'Len',
'Retransmission', 'Duplicate', 'Lost_packets']

# Separate features and the target value
x = df.loc[:, features].values

# Feature standardization
x = StandardScaler().fit_transform(x)

#Define PCA for 4 comopnents
pca = PCA(n_components=4)
#Fit the data to the model
principalComponents = pca.fit_transform(x)

#Print principal components' variance ratios
print("Variance ratio")
print(pca.explained_variance_ratio_)

#Add PCA parameter variance results to new dataframe and print it
dataset_pca = pd.DataFrame(abs(pca.components_), columns=['Delay',
'Length', 'Seq', 'Ack', 'Win', 'Len', 'Retransmission', 'Duplicate',
'Lost_packets'], index=['PC_1', 'PC_2', 'PC_3', 'PC_4'])
print(dataset_pca)

#Print eigenvectors and eigenvalues
print('\neigenvectors’,pca.components_)
print(‘\neigenvalues’,pca.explained_variance_)
```

Figure 42. PCA source code.

Appendix 7 – K-means source code

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

#Open the dataset and load it to the dataframe
url =
'https://raw.githubusercontent.com/kaarel2/test/main/RPi_Traces/Combin
ed3_retransmission.csv'
df = pd.read_csv(url, names=['Delay', 'Length', 'Seq', 'Ack', 'Win',
'Len', 'Retransmission', 'Duplicate', 'Lost_packets',
'cat_technology'])

#Define features
features = ['Delay', 'Length', 'Seq', 'Ack', 'Win', 'Len',
'Retransmission', 'Duplicate', 'Lost_packets']

# Separate features from the target value
x = df.loc[:, features].values

# Separate the target value
y = df.loc[:,['cat_technology']].values

# Feature standardization
x = StandardScaler().fit_transform(x)
scaled_x = x

#Define arguments for k-means model
kmeans_kwargs = {"init":"random", "n_init": 10, "max_iter":300,
"random_state": 42}

#Calculate SSE for different number of clusters
sse = []
for k in range(1, 8):
   kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
   kmeans.fit(scaled_x)
   sse.append(kmeans.inertia_)

#Plot SSE vs number of clusters
plt.plot(range(1, 8), sse)
plt.xticks(range(1, 8))
plt.ylabel("SSE - Sum of the Squared Error")
plt.xlabel("Number of Clusters")
```

```
plt.show()

#Perform k-means analysis for 6 clusters
kmeans_clusters = KMeans(n_clusters=6, **kmeans_kwargs)
kmeans_clusters.fit(scaled_x)
#Print kmeans parameters
print('k-means SSE',kmeans_clusters.inertia_)
print('centroid locations',kmeans_clusters.cluster_centers_)
print('Iterations to converge',kmeans_clusters.n_iter_)

#Print kmeans labels to file
df_results = pd.DataFrame(kmeans_clusters.labels_)
with pd.ExcelWriter('kmeans_labels.xlsx', engine='xlsxwriter') as
writer:
    df_results.to_excel(writer, sheet_name='Sheet1')
```

Figure 43. k-means source code.

Appendix 8 – Linear regression parameters calculation source code

```
import pandas as pd
from sklearn.linear_model import LinearRegression

#Open file with data, load to dataframe
df2 = pd.read_excel('Regression_data.xlsx', sheet_name = 'Leht2')

#Define linear regression models
model2 = LinearRegression()
model3 = LinearRegression()
model4 = LinearRegression()
model5 = LinearRegression()
model6 = LinearRegression()
model7 = LinearRegression()
model8 = LinearRegression()
model9 = LinearRegression()

#Define dependent, independent variables
X2, y2 = df2[['Count']], df2.No_of_devices
X4, y4 = df2[['Average']], df2.No_of_devices
X5, y5 = df2[['Median']], df2.No_of_devices
X6, y6 = df2[['dup_count']], df2.No_of_devices
X7, y7 = df2[['retrans_count']], df2.No_of_devices
X8, y8 = df2[['Count','Median','retrans_count']], df2.No_of_devices
X9, y9 = df2[['Count','Median','dup_count','retrans_count']],
df2.No_of_devices
X3, y3 = df2[['Count', 'Average', 'Median', 'dup_count',
'retrans_count']], df2.No_of_devices

#Fit models for different variable combinations
model2.fit(X2, y2)
print('\nCount of RTT intercept, coefficient, score')
print(model2.intercept_, model2.coef_, model2.score(X2, y2))
model4.fit(X4, y4)
print('\nAverage of RTT intercept, coefficient, score')
print(model4.intercept_, model4.coef_, model4.score(X4, y4))
model5.fit(X5, y5)
print('\nMedian of RTT intercept, coefficient, score')
print(model5.intercept_, model5.coef_, model5.score(X5, y5))
model6.fit(X6, y6)
print('\nCount of duplicate packets intercept, coefficient, score')
print(model6.intercept_, model6.coef_, model6.score(X6, y6))
```

```
model7.fit(X7, y7)
print('\nCount of retransmission packets intercept, coefficient,
score')
print(model7.intercept_, model7.coef_, model7.score(X7, y7))
model8.fit(X8, y8)
print('\nCount, Median of RTT; count of retransmission packets
intercept, coefficient, score')
print(model8.intercept_, model8.coef_, model8.score(X8, y8))
model9.fit(X9, y9)
print('\nCount, Meidan of RTT; count of duplicate and retransmission
packets intercept, coefficient, score')
print(model9.intercept_, model9.coef_, model9.score(X9, y9))
print('\nAll parameters intercept, coefficient, score')
model3.fit(X3, y3)
print(model3.intercept_, model3.coef_, model3.score(X3, y3))

#Create dataframe for printing of regression models' results
d = {'Intercept': [model2.intercept_, model4.intercept_,
model5.intercept_, model6.intercept_, model7.intercept_,
model8.intercept_, model9.intercept_, model3.intercept_],
'Coefficient(s)': [model2.coef_, model4.coef_, model5.coef_,
model6.coef_, model7.coef_, model8.coef_, model9.coef_, model3.coef_],
'Score': [model2.score(X2, y2), model4.score(X4, y4), model5.score(X5,
y5), model6.score(X6, y6), model7.score(X7, y7), model8.score(X8, y8),
model9.score(X9, y9), model3.score(X3, y3)]}
df_excel = pd.DataFrame(data = d, index = ['Count of RTT', 'Average of
RTT', 'Median of RTT', 'Count of duplicate ACKs', 'Count of
retransmission packets', '1,3,5 parameter', '1,3,4,5 parameter', 'All
parameters'])

#Print regression parameters' results to excel file
with pd.ExcelWriter('summary_regression.xlsx', engine='xlsxwriter') as
writer:
    df_excel.to_excel(writer, sheet_name='Sheet1', startrow = 1,
float_format = "%0.2f")
    worksheet = writer.sheets['Sheet1']
    worksheet.write(0, 0, 'Regression analysis for different
combinations')
    writer.sheets['Sheet1'].set_column(0, 0, 29)
    writer.sheets['Sheet1'].set_column(2, 2, 72)
```

Figure 44. Linear regression parameters calculation source code.

Appendix 9 – Non-exclusive licence for reproduction and publication of a graduation thesis[3]

I Kaarel Koovit

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Machine Learning Based Modelling of TCP Data on Raspberry Pi”, supervised by Prof. Yannick Le Moullec
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

02.01.2024

3 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.