CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Data-driven configuration recommendation for microwave networks

A comparison of machine learning approaches for the recommendation of configurations and the detection of configuration anomalies

Master's thesis in Computer science and engineering

Simon Pütz
Simon Hallborn

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Data-driven configuration recommendation for microwave networks

A comparison of machine learning approaches for the recommendation of configurations and the detection of configuration anomalies

Simon Pütz
Simon Hallborn

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Data-driven configuration recommendation for microwave networks
A comparison of machine learning approaches for the recommendation of configurations and the detection of configuration anomalies
Simon Pütz, Simon Hallborn

Supervisor: Marwa Naili, Department of Computer Science and Engineering
Advisor: Martin Sjödin and Patrik Olesen, Ericsson AB
Examiner: John Hughes, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Data-driven configuration recommendation for microwave networks
A comparison of machine learning approaches for the recommendation of configurations and the detection of configuration anomalies
Simon Pütz
Simon Hallborn
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

As mobile networks grow and the demand for faster connections and a better reachability increases, telecommunication providers are looking ahead to an increasing effort to maintain and plan their networks. It is therefore of interest to avoid manual maintenance and planning of mobile networks and look into possibilities to help automate such processes. The planning and configuration of microwave link networks involves manual steps resulting in an increased effort for maintenance and the risk of manual mistakes. We therefore investigate the usage of the network's data to train machine learning models that predict a link's configuration setting for given information of its surroundings, and to give configuration recommendations for possible misconfigurations. The results show that the available data for microwave networks can be used to predict some configurations quite accurately and therefore presents an opportunity to automate parts of the configuration process for microwave links. However, the evaluation of our recommendations is challenging as the application of our recommendations is risky and might harm the networks in an early stage.

Keywords: Configuration, Recommendation, Machine learning, Microwave network

Acknowledgements

We would like to express our gratitude to our supervisors Patrik Olesen and Martin Sjödin at Ericsson, for their counseling and guidance that greatly helped the thesis come together. Also, we would like to thank our supervisor Marwa Naili for her support and engagement during the process of the thesis. Furthermore, we would also like to thank Björn Bäckemo at Ericsson, for his enthusiasm in extending our knowledge of the field.

Simon Pütz, Gothenburg, July 2020
Simon Hallborn, Gothenburg, July 2020

Contents

List of Figures

List of Tables

List of Theories

1
Introduction

Microwave links are part of a telecommunication's backhaul network and one way to transmit data between radio base stations and the core networks of providers. Other options to transmit data are fiber optics or copper wires and are typically mixed with microwave links in a provider's backhaul network. Microwave links are commonly used as installation time is short and cables do not have to be buried. However, location and configuration of links within a microwave network need to be planned carefully to meet the requirements for reachability, capacity and a provider's regulations, such as the frequencies a provider is entitled to use. Planning link locations and configurations as well as the network's maintenance are therefore important tasks to reach the performance goals of a telecommunication network.

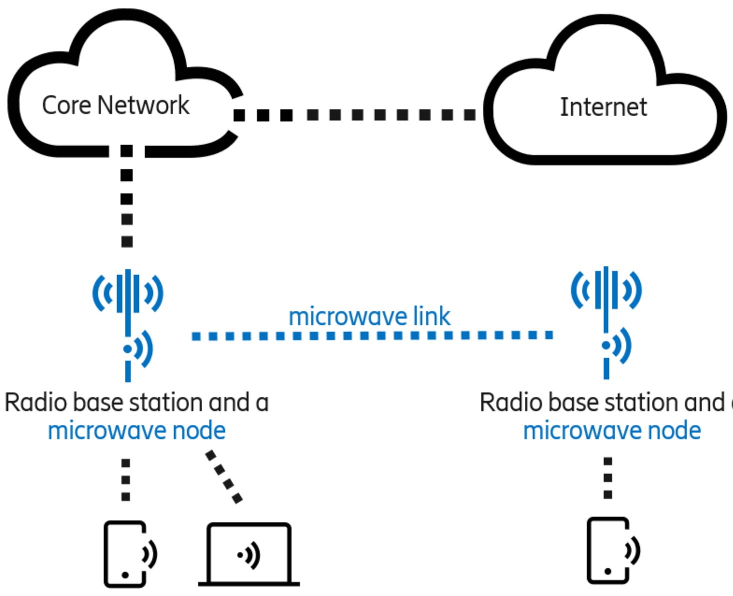


Figure 1.1: Radio base stations communicate with mobile end devices. A microwave link is built between two microwave nodes. Microwave links transmit data between the radio base stations and access points of the core network, either directly or via hops among other microwave links. Only some microwave nodes are connected to the core network directly.

Each microwave link is planned considering information about a link's context, such as involved hardware and software information, weather conditions of the node locations or the link's length. The current workflow for planning the configuration of

microwave links is assisted by a variety of planning tools that consider this information and further requirements for the link. The process is not fully automatic and requires manual steps. This results in a degree of freedom inside the planning tools and the actual application of configurations, allowing mobile network providers to configure microwave links to their special needs. With manual work comes the risk of human mistakes and with that the risk of misconfigurations for microwave links. A misconfiguration can lead to a microwave link operating with a performance below its potential.
The demand for mobile communication has increased exponentially since the start of the twenty-first century. This demand is met by pushing new generations of mobile phone technologies and each new generation increases the network complexity as networks become more heterogeneous e.g. by an increasing diversity of services and devices. As mobile networks grow and the demand for faster connections and a better reachability increases, telecommunication providers are looking ahead to an increasing effort to maintain and plan their networks. Looking at the upcoming generation of telecommunication technologies, 5G, higher bandwidth and low latency will be requested. A key driver for this development is the expectation of an increasing number of Internet-of-Things (IoT) devices, implying an increase of subscriptions per client in the future.
On the other hand, telecommunication companies like Ericsson are developing ways to automate planning and maintenance processes with the goal of self-organized and fully automated networks (SON) [19] that are supposed to tackle these upcoming challenges. A tool for identifying misconfigurations and providing configuration recommendations could contribute to automating configuration processes of microwave links and therefore assist telecommunication providers along the evolution of telecommunication networks.

1.1 Problem statement

Suitable configuration settings are based on the characteristics of a link and its environment. As this dependency is used in the current workflow, the given data for configuration and link information is supposed to reflect this relationship. We can then formulate the following problem:

Consider a microwave link with a vector $\bar{x} = (x_1, x_2, \ldots, x_n)$ of configuration settings, e.g. upper and lower bound of the QAM constellations, which are a family of modulation methods for information transmission, and the link's transmitting power. Consider $\bar{y} = (y_1, y_2, \ldots, y_m)$ to be a vector of observable variables describing the context of a link, e.g. information about hardware, software and location. Given this information, we aim to develop a machine learning model that learns the relationship between the configuration and context. With this model we can predict whether a configuration setting for a microwave link, given the context, is configured like other similar microwave links. Hence, this model can act as an anomaly detection and is able to locate candidates for misconfigurations. Assuming that misconfigurations make up a small percentage of the data, our model might be able to detect misconfigurations for microwave links and recommend suitable configuration settings for each misconfigured microwave link.

1.2 Related Work

Previous research about the optimization of microwave links configurations mainly focuses on optimizing network performance by considering only a few configurations and the dynamic interaction of their settings. Turke and Koonert [30] investigated the optimization of antenna tilt and traffic distribution in Universal Mobile Telecommunications Systems (UMTS) by applying first a heuristic-based optimization followed by a local-search-based optimization. The respected configuration features were updated dynamically and not in a static scenario as ours. Also the optimization problem is narrowed down to the interaction of settings for two configuration features and does not cover the configuration optimization for all important configuration features of a radio link. The two configuration features and their respective optimization targets represent a rather specific approach for microwave link optimization. Awada et al. [4] propose an approximate optimization of parameters within a Long Term Evolution (LTE) System using Taguchi's Methods in an iterative manner. The optimization is focusing on a few configuration features, namely an uplink power control parameter, tilt of the antenna and the azimuth orientation of the antenna. The optimization takes place in a network-planning environment and therefore only simulates the outcomes of configuration changes. Islam et al. [31] have a similar approach as they optimize antenna tilt and pilot power using heuristic algorithms.

As mentioned before SON are on the rise and previous research underlines the potential for data-driven approaches such as machine learning to enhance network performance [20, 22]. However, there does not seem to be any previous research considering machine learning approaches for the recommendation of radio network configurations. This opens up a research gap, as microwave link networks provide a collection of live data and therefore in principle allow exploring data driven approaches.
This brings up the following research questions that will be answered in this master thesis project:

1. Is the utilization of machine learning models a feasible methodology to configure microwave networks?
    1.1. Are our models capable of identifying misconfigurations as outliers?
    1.2. Can our models provide configuration settings that improve a link's performance?

1.3 Roadmap

We integrated necessary and applied concepts inside the method section. For each theory or concept we provide a theory box to clarify the main ideas of it. This replaces the more common theory section of a thesis paper. The method section itself is dedicated to the procedures that we performed, starting with the data processing and feature selection and ending in different ways of evaluating our results. We tried to organize this section in chronological order to facilitate retracing the sequence of our pipeline. We conclude our thesis by evaluating experiments with different data sets, anomaly detections, clustering algorithms and machine learning models. These results are put into perspective and are discussed. Afterwards, we talk about possibilities for future work. The appendices presents useful tables and plots for a deeper understanding.

1.4 Ethical considerations

This thesis project investigates the possibilities of monitoring and replacing manual work of microwave link configurations. We have to keep in mind that some results of this thesis could lead to the reduction of manual steps and therefore the reduction of job positions in that area.
Possible failure of a system should always be considered when applying machine learning techniques. Applications involving machine learning techniques tend to replace or support complex processes such as important decision making. We should also keep in mind that a model is never perfect and bears the risk of causing harm when e.g. making poor predictions. In our specific case the application of poor recommendations for configuration settings in a microwave network could lead to a drop of performance of the whole telecommunication network or even a crash.

2 Methods

In this thesis we propose a model that addresses the stated problem and respects the specific characteristics of the data as well as the following assumptions. We assume that the current process of setting configurations is already quite accurate as a lot of expert knowledge and tools are involved. The main assumption is therefore that most configurations are fine for given environments, and only a few misconfigurations exist in the data. As these misconfigurations appear less frequently compared to regular configurations, we expect them to act as outliers within the data. A classification model for a configuration feature should therefore be insensitive to such outliers resulting in high accuracy on correct configuration settings and low accuracy on wrong configuration settings. Mispredictions of the model then result in recommendations for these configuration features. We highly rely on these assumptions as a ground-truth for the correctness of configuration settings in the data is not given, implying an unsupervised classification problem.

One way to design a model that is insensitive to uncommon configuration settings is to leave out outliers in the training data. This can be achieved by splitting up the data by means of an anomaly detection. We first cluster the data using the environmental features. This results in clusters that are quite close in terms of environmental features and should have a small variance of configuration settings. Finding outliers then becomes a one-dimensional anomaly detection problem for each configuration feature in each cluster. Uncommon configuration settings within one cluster are then considered outliers.

Our approach is centered around a Gradient Boosting model for each configuration feature. The recommendation task is reformulated as a classification task where a misclassification could be interpreted as a recommendation for a different configuration setting. To achieve this we first use non-anomalies to train and test the models, aiming for good predictions. In the second step we apply the trained models on the anomalies.

We then try to evaluate the recommended configuration settings using a performance metric. The metric is supposed to be based on data describing different aspects of a link's performance such as modulation states and number of dropped packages over time. As this information is not given for the recommended configuration settings we have to approximate the performance metric by finding existing links with similar environments and configuration settings.

The described attempt resulted in the process that is visualized in Figure 2.1. The figure is meant to be an overview and each step is described in further detail in the coming sections.

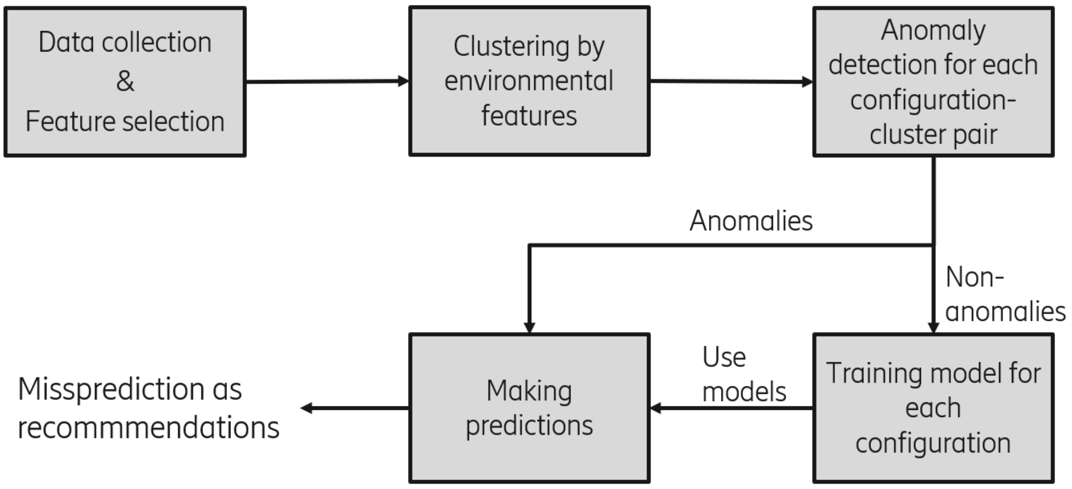

Data collection
&
Feature selection
Clustering by environmental features
Anomaly detection for each configuration-cluster pair
Anomalies
Non-anomalies
Training model for each configuration
Use models
Making predictions
Missprediction as recommmendations

Figure 2.1: Flowchart describing the whole process of obtaining recommendations, starting from the data collection and ending with the recommendation of new configuration features and their evaluation by experts. The anomaly detection and training process are applied for each configuration feature. This results in a model and set of anomalies for each configuration feature.

2.1 Feature engineering

It is quite common for machine learning projects to have a lot of emphasis on feature engineering as machine learning tools rely on clean data and a selection of meaningful features that can actually explain the desired relationship. In our case this involved: a) Investigation of the data's meaning, as an explanation for the data was not given initially and a strong domain knowledge was required to create meaningful connections between different information. b) The data had to be cleaned carefully, specially because some notations do not correspond among providers and links. This appeared in the form of different categories, e.g. *QAM-4-std* and *qam 4*, both describing the modulation state QAM 4 without protection, or differences for the notations of radio terminal ids among different providers. This resulted in a careful study of feature and target candidates, their value ranges and associations among each other, driven by the guidance of domain experts.

2.1.1 Data retrieval

The data sources used in our project are fairly unexplored and new to Ericsson. We mostly dealt with raw data and could only rely on some data cleaning and pre-processing from previous projects at Ericsson. We obtained classification data describing signal interference and meta data such as a microwave node's location from an Influx time series database. Other data was obtained from an Impala database that stores live data from radio networks of certain customers. Information is distributed over several tables and is described with technical and shortened names. With the support of domain experts we explored the data and ended up with a selection of suitable features for our application as a result of multiple iterations

of feature selection, inquiries of experts and the evaluation of our models.
Figure 2.2 shows the relationship between the different database tables we ended up using. Each box represents a table within a database. The links and cardinalities between the tables show the relationship between the data and how it is connected. The data and its connections are explained in the following, starting from the hardware table *raw_hw* to the meta data table *link_meta*.

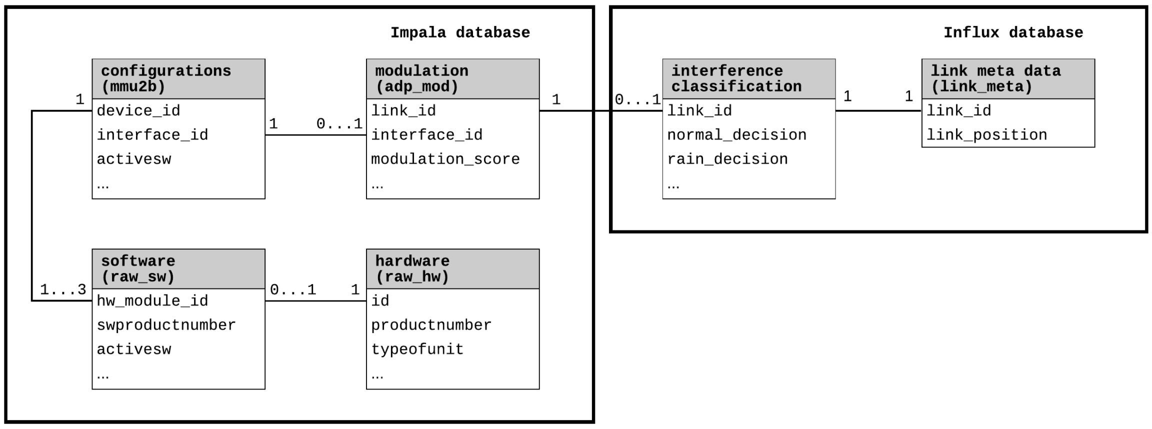


Figure 2.2: Entity-relationship diagram showing the relationship between the different data sources being used.

raw_hw includes hardware information such as product number for each device that is listed. The given id in *raw_hw* matches with a given id *hw_module_id* in table *raw_sw*. Not every device requires software, which is why software information is not given for some devices. We do not require information for all devices as only a few contribute to a radio link. Therefore, we focused on radios, modems and node processor units. The given id *hw_module_id* consists of the node id, information about the position within a physical rack and an indicator for the type of device. For a modem this id might look like 1000_1/3_M. We split up and combine information about devices that belong to the same link. To do so, we have to understand how a link is built up physically: each node has one node processor unit and one fan unit. A node is usually involved in multiple links. Each link then consists of a radio-modem pair. A modem can be part of multiple links, a radio however is always part of only one link. In most cases a link only consists of one radio on each side (1+0 link). There are rare cases where a link can consist of two or more radios on each side. These cases are for example called 2+0 or 1+1 links and are excluded in this project as irregularities in the data made it difficult to include them. Figure 2.3 illustrates these relationships. Combining the information according to the devices that are involved in each link results in hardware and software information on a link level.

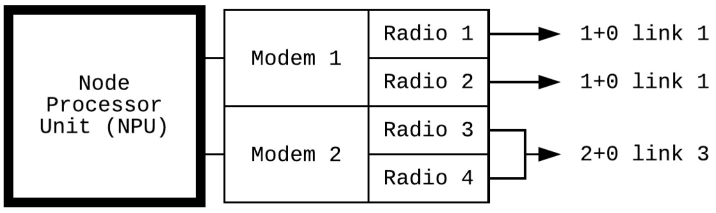


Figure 2.3: The illustration shows how different devices within one node are involved in either a 1+0 or 2+0 link. The figure shows how a modem can be part of multiple microwave links, a radio is always part of only one link.

As we ignore anything but 1+0 links we can identify a link using the id of its radio. We can then create an id of the same shape for every column in the table *mmu2b* and aggregate the information. *mmu2b* provides information on a link level, such as modem and radio configuration features and further information about modem and radio, such as base frequencies and capabilities for the capacity. Information about the modulation states a link has been in over time is given by the table *adp_mod*. Information about the radio is not given in this table so we have to connect the data using the id of a link's interface. Modulation data is not necessarily given for every link as measuring times spent in different modulations might not be enabled or supported by some links. Therefore, some links are disregarded when considering the modulation data. The use of the modulation data is further described in Section 2.3.1. The process so far only includes data from the Impala database. In addition we want to include data that is stored in an Influx database and is part of an application dealing with the classification of signal fading events. This includes the classification decisions over time as well as meta data such as the microwave node locations. For a longer period of time this data shows what events typically affect the operation conditions of a microwave link. The data has a different identifier for microwave links, namely *link_id*. Fortunately the table *adp_mod* in the Impala database includes enough information to construct this *link_id*. In a later stage of the project we discovered that the information needed to construct *link_ids* is also given in the table *mmu2b*, but requires some provider-specific processing as notations among providers vary. We can then connect classification data and link meta data with data from the Impala database. The scope of the application for the classification of signal fading events is however limited as the data required for the classification task is only provided for a selection of links, mainly in urban areas. As a result we exclude 50 % of the links as we decide to include this additional data.

2.1.2 Cleaning data

The obtained data is not ready for any further steps yet and needs to be cleaned before proceeding. In some cases there are different values describing the same subject, possibly due to changing norms over time or different norms between different hardware and software. This also includes different notations for missing values such as "NaN". These cases are identified by investigating value ranges of the feature candidates and are aligned again after consulting experts.
Other features represent numeric values but are obtained in categorical form. To

avoid unnecessary encoding before training the models, as some Gradient Boosting models require a numerical input, we can transform these values back into a numerical form by stripping apart parts of the entries and converting them into numerical data types.
Some categorical features consist of compressed information that can be stripped apart to allow the models to find relationships with this more detailed information. As an example, the product number for radios includes information about the radio's family, the frequency index within this family, whether this is a high power radio, whether this is an agile radio and in which half of the frequency band this radio is operating on. Stripping this information apart does not necessarily improve the accuracy but can lead to more reasonable recommendations, cleaner features and allows us to remove redundant information among features. This results in a smaller range of values for categorical features and helps to prevent feature representations of categorical data, such as one-hot encoding, to blow up in dimensions. High encoding dimensions can increase the training time to a critical level for some classifiers. As the classifier from CatBoost includes a specific representation of categorical features, reducing the range of categories is crucial to enable training in a reasonable time.
Some features represent attributes that come in a wide and detailed numerical value range. As we define the configuration recommendation as a classification problem the models will struggle to predict the exact value. For configuration features of that shape a regressor could be used instead of a classifier. We noticed that it leads to problems when evaluating the models and bringing together the results in the end. This is because classifiers are evaluated using accuracy and regressor using errors. As we train models for several configuration features for each classifier in our scope it becomes more difficult, but not impossible, to compare the model evaluations. Sticking to a classification problem makes the broad exploration of multiple configuration features and classifier implementations easier for us, but for more precise recommendations a mix of classifiers and regressors could be applied in the future. Therefore we decided to transform numerical values with a high range of values into ordinal categories by applying bins with equal ranges over the whole value range of the feature. We accept losing some precision here as we group values into bins in order to enable the application of a classifier that still provides good recommendations for configuration settings that vary too much from a more common state.

2.2 Selection of configuration and environmental features

The selection of features is crucial for the success of our models. Essentially, this consists of the feature selection for configuration features and environmental features. The selection of configuration features is straightforward as it determines the scope of our application. However, we have to see whether we have all the information needed to make good predictions for each configuration feature. The selection of environmental features is motivated by finding all the data that is necessary to explain the variance in the values of our selected configuration features. This process was driven by the evaluation of association matrices, expert inquiries and a general exploration of the data. Association matrices are used to express correlations between categorical data. One metric to quantify the correlation between categorical features is Cramer's V [8] by Cramér, Harald.

Theory 1: Cramer's V

Theory 1
Cramer's V is a test that extends the Pearson's Chi-squared test. The chi-squared test starts by taking a sample of size n from two categories A and B for $i = 1, \dots, r$; $j = 1, \dots, k$, where k is the number of columns and r is the number of rows. Let the frequency $n_{i,j}$ be the number of times A_i, B_j were observed. The Chi-squared statistic is then defined by

$$X^2 = \sum_{i,j} \frac{(n_{i,j} - \frac{n_{i.}n_{.j}}{n})^2}{\frac{n_{i.}n_{.j}}{n}},$$

where $n_{i.}$ is the number of category values of type i ignoring the column attribute and $n_{.j}$ is the number of category values of type j ignoring the row attribute. Cramer's V is then defined as

$$V = \sqrt{\frac{X^2/n}{min(k-1, r-1)}},$$

which returns a value between $[0, 1]$ where 0 means no association and 1 means maximum association between A and B.

A high association between two environmental features or two configuration features indicates redundant information, whereas a high association between an environmental feature and a configuration feature indicates that the environmental feature is capable of explaining the variance of the configuration settings. We plotted these association matrices as association heatmaps which are equivalent to correlation heatmaps. These plots can be found in Appendix A.
The process of finding the right set of configuration features is mainly relying on experts stating which configuration features have the biggest impact on a link and which configuration features actually can be changed. We then identified and removed redundant configuration features using association matrices. The final set of

configuration features includes preferred modulations, maximum modulations, channel spacing, various configuration features dealing with input and output power as well as alarm thresholds. In a late stage of the project we removed a few configuration features as their models performed poorly, indicating that we lack additional environmental features to predict these configuration features. A complete list of considered configuration features and their meaning is listed in the Appendix B.
The process for choosing environmental features consists of identifying and removing redundant information and identifying feature importance using association matrices and a simple Gradient Boosting model. These models are often used to investigate feature importance and are capable of ranking features based on the impact they have on the decision making. However, the environmental feature selection is not important for training tree-based models in general as they are robust to unnecessary features. Narrowing down environmental features is still important for the evaluation of results as the reduction of features can lead to better results for the nearest neighbour search using a kd-tree. More information about this is given in Section 2.9. Candidates for environmental features mainly consist of information about software and hardware of a link and some additional attributes such as base frequencies, temperature, operating conditions classification, climate zone classification according to the location of a microwave node and a link's length.
Figure 2.4 gives an example of value distributions for different numerical and categorical environmental features we included in our models. Figure 2.5 gives an example of a categorical and numerical configuration feature and their value distributions. All graphs show histograms of the cleaned up data. Figure 2.5 gives an example of how the configuration settings, or classes of the configuration feature, can be quite imbalanced. We have to respect this when training and evaluating the model. This is taken into further account in Sections 2.8.1 and 2.8.3.

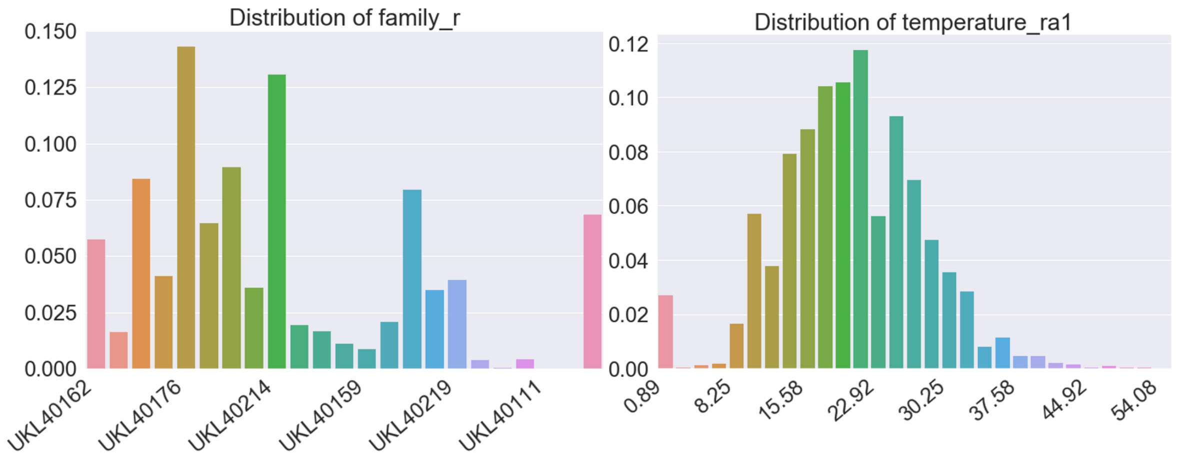


Figure 2.4: Histograms showing the value distributions of the categorical feature 'radio family' (left) and the numerical feature 'radio temperature' (right).

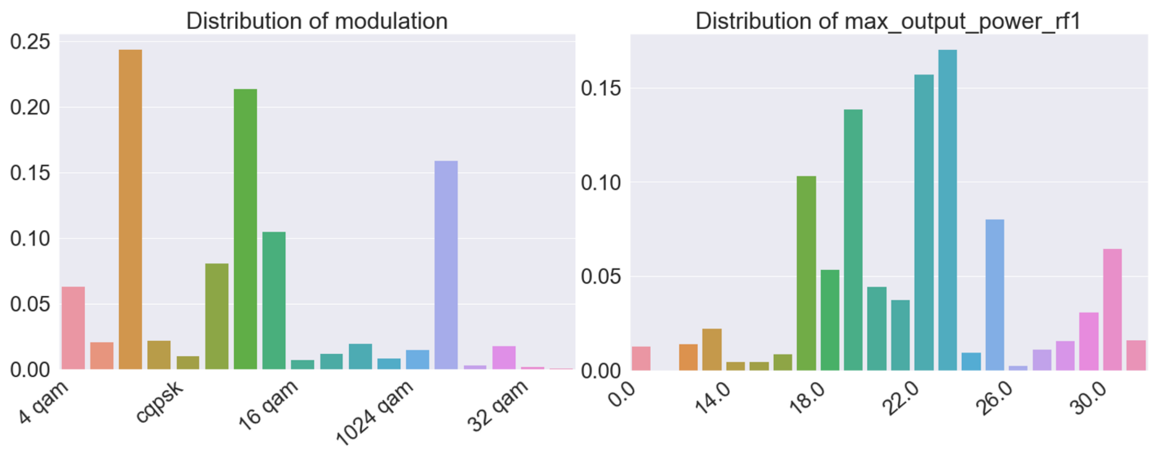


Figure 2.5: Histograms showing the value distributions of the categorical configuration feature 'modulation' (left) and the numerical configuration feature 'maximum output power' (right).

2.3 Performance investigation

During the thesis, we investigated different data sources to explain the performance of a microwave link's operating condition. The performance could be an analytical function whose parameters are part of different available data sources. The different data sources were investigated based on the consultation from experts. For each data source there are counters of live data, explaining different aspects of performance. The counters were obtained by aggregating the data over a given interval. We wanted the interval to span at least a couple of months, such that we obtain a large data set. Since the counters are obtained from live data, it is also important to aggregate over a longer period so that the variance in the counters is reduced. For example, during rainy seasons some counters experience a larger deviation to its expected value than on dry seasons. We ended up using an interval over six months, but this interval can be extended for future purposes. We were not able to implement this function and showcase it in the result section, but we will show our investigation beginning with the counters that were considered.

2.3.1 Performance counters

We obtained the counters from two data sources: traffic and modulation data. Other performance counters such as the bit error rate and packet error rate were not available for our project. Based on feedback from experts, we considered using the number of dropped packages (dp) from the traffic data and 31 different QAM constellations from the modulation data. Briefly, QAM constellations are a family of modulation methods for information transmission. From these counters we wanted to determine if a microwave link's operating condition had been good or not.

Based on previous work, we used Equation 2.1 that ranks each microwave link in terms of how much it deviates from its expected modulation. This equation considers the relative time a microwave link has spent in a certain QAM constellation and how far away the QAM constellation is from its expected level. The function in Equation 2.1 was obtained from empirical testing and using visualization tools to evaluate the result. The modulation score, ms is defined as follows

$$ms = \sum_{i=0}^{n} \frac{1}{\sqrt{\frac{t_i}{t_{tot}}}(|n-r|+1)}, \tag{2.1}$$

where n is the number of QAM constellations, r is the expected QAM constellation over the interval, t_i is the time a microwave link has spent in the i^{th} QAM constellation and t_{tot} is the total time the microwave link has been sending data during the interval.

Since we already had the modulation score we considered creating a traffic score for the dropped packages dp. A possible equation for the traffic score is

$$ts = \frac{1}{\log(dp)+1}. \tag{2.2}$$

Equation 2.2 could be tested empirically and be investigated further and adjusted using visualization tools. However, when aggregating the data we noticed that the number of microwave links with both traffic and modulation data was less than we expected. This is because only some microwave links measure both counters. Hence, we had to leave the performance function for future work. The main idea behind it is explained in the next Section and some ideas for the usage of it are explained in Section 5.1.

2.3.2 Performance function

The main idea behind the performance function was that we wanted a measurement to determine which microwave link's operating conditions were doing well. Since customers define a good operating condition differently, we would have considered this by creating an individual performance function P_c for each customer c

$$P_c = f(ms_c, \dots, ts_c), \tag{2.3}$$

where ms_c and ts_c are the modulation and traffic scores from the data sources of customer c. For a final performance function, more performance measurements in the fashion of the modulation score are required. The function itself including weights etc. needs to be created empirically.

2.4 Feature representation

All models require numerical input which forces us to encode categorical data into numerical values. There are different ways to do so as some encodings will create an ordinal structure, such that the values of the feature representation has an internal structure (for example being able to sort them based on magnitude). One example of an ordinal encoding is label encoding.

Theory 2: Label Encoding

Theory 2
Label encoding assigns a value between 0 and $N-1$ to each instance of a feature with N unique values. The result is a one-dimensional numerical representation of the feature. As this encoding approach assigns values in a certain order, as one value is smaller or bigger than another one, this encoding can be problematic for nominal categorical data. Nominal categorical data consists of categories that do not have a particular order like for example names of countries. Label encoding will apply a certain order to this value range that is not meaningful and can be picked up by some models. For other models, such as tree-based models, this approach is valid as a feature representation. One advantage of using label encoding is its resource efficiency as features are always represented in one dimension.

Other encodings such as nominal encoding, eliminates any internal structure such that magnitude-based sorting is not possible. One example of a nominal encoding is one-hot encoding.

Theory 3: One-hot encoding

Theory 3
One-hot-encoding is a simple and intuitive encoding that maps each category value to a vector with dimension $[n, 1]$, where n are the unique category values. This creates a $[n, m]$ matrix for the m category values that needs to be mapped. This mapping does not consider the distance between vectors inside the matrix, since the scalar product between the vectors are 0. It considers the category values as nominals, so the order of the category is disregarded. The matrix is also very expensive for storage and computation if m and n are large, which is normal in applications like natural language processing.

Gradient boosting models do not pick up ordinal structures as they do not apply weights such as neural networks do. Hence, the feature representation for the model is not important and we used label encoding as a representation since it is less expensive than one-hot encoding.

Since we wanted a model that was robust to outliers, we wanted to identify outlying configuration settings and separate them from test and training. To identify an outlier we needed an encoding that tries to express some kind of closeness between different values. We decided to use the model Cat2Vec [33] by Wen et al. This decision was based on the result from the result of the paper that showed how well the model was performing in comparison to other techniques. This was the main reason to why we decided to use this model as a feature representation to span distance between different configuration settings. Cat2Vec is also an extension of Word2Vec [25] by Mikolov et al, which we were both quite familiar with.

Theory 4: Word2Vec

Theory 4
Word2Vec trains a neural network on a set of given sentences and uses the learned weights of the hidden layers as a representation of words. The representation is based on the network architecture which varies depending on the type of method that it utilizes. There are two methods that are available (CBOW) Continuous Bag of Words and Skip-gram. Using Continuous Bag of Words creates a classification task to predict the next word in a given sentence for example 'This was absolutely _'. Skip-gram creates a classification task to predict neighbouring words to a given word. For example, by finding the three previous words in the sentence '_ _ _ delightful'.

To determine the architecture of Word2Vec, we had to decide between CBOW or Skip-gram. The main issue with CBOW is that the predicted words will be influenced by similar ones, which means if the true word is rarely occurring, the model will likely predict something more common. The main issue with Skip-gram is the time complexity, because it needs to train on more data compared to CBOW.
The main benefit of CBOW is that common words are represented well and for Skip-gram that it is good in representing both common and rare words by considering the context. The latter was shown in [25] and in the original paper both by Mikolov et al. To compare the two methods, we looked at the paper [14] by Guthrie et al, and the conclusion was that Skip-gram and CBOW are similar in terms of effectiveness. Considering this we favoured Skip-gram, mostly to ensure that the model would represent rare words effectively. The added time complexity was not a huge concern to us since we believed that a large portion of the outliers would consist of these rare words.

Theory 5: Cat2Vec

Theory 5
Replace all category values by string values of $'c+v'$ where c is the category and v is the category value. Afterwards, every string value of $'c+v'$ is mapped to a dictionary. This guarantees that all the words in the dictionary are different since all the category names are unique.

Then each instance of the data is made into a sentence, so for example instance i would yield the sentence $[c_0 + v_i, c_1 + v_i, ..., c_n + v_i]$ where n is the number of categories and each word is separated with a comma. The sentences gets shuffled and are used to train a Word2Vec model that learns similarities between words and creates a distance space by predicting the context of a given category value from the created dictionary.

We suspected that there could be unnecessary features that would not contribute much to our model. Hence we considered utilizing dimensionality reduction methods to remove the unnecessary features from the data set.

2.5 Dimensionality Reduction

For systems with a lot of parameters it can be troublesome to span a distance space that explains distance between points. This could be when some of the parameters are heavily correlated or are just adding noise to the distance space. Even after a careful feature selection, there might just be too many features left. According to [10] by Ding and He, when reducing the dimensionality of their distance space, it lead to improved results. They tested the technique Principal Component Analysis (PCA), which is commonly used for dimension reduction. Based on the review [32] from 2009 by Van der Maaten et al. they tested different dimensionality reduction techniques on different data sets and came to the conclusion that Principal Component Analysis (PCA) was the one that performed the best.

Theory 6: PCA

Theory 6
PCA projects a n dimensional space, that is obtained from some initial data set X_n, into a smaller dimensional space X_m. As the dimension of the space is reduced, the information-loss is considered such that the subspace does not lose too much information. The consideration of the information-loss is done in several steps, where the first step is to standardize the data by subtracting the mean and dividing by the standard deviation for each value of each variable. This is done to scale the data so that no extreme points will severely affect the mapping of the data. The second step is calculating the covariance matrix for the scaled data, X'_n that summarizes the correlation for all the pairs of variables. From the covariance matrix the eigenvalues and eigenvectors are calculated, and by ranking each eigenvector based on the corresponding eigenvalue and selecting the m largest values, there will be m vectors left. The first of these vectors is the eigenvector that has the most amount of information in terms of variance, which is also known as the principal component of the greatest significance. By stacking the principal components in the order of the significance as columns of a matrix, it will create a feature matrix F. From Equation 2.4 the new data

$$X_m = F^T * X'_n, \tag{2.4}$$

is obtained that has m dimensions instead of n.

We wanted to ensure that we chose the right method for the feature representation. Hence, we applied PCA to reduce the number of dimensions and kept at least 95% of the sum from the variance of the principal components for both Skip-gram and CBOW. Once we obtained the principal components, we first thought of directly applying anomaly detection methods to both the environment and the configurations to find configuration values that were deviating from the norm. We realized that this would not yield a good selection of anomalies since there were a lot of variance inside the environment parameters. Hence, we thought of first clustering the environment parameters and afterwards apply anomaly detection inside each environment cluster to find deviating configuration settings for each configuration feature.

2.6 Clustering

Clustering is utilized to find clusters of groups in a data set where the contained data points are similar. Similarities inside a cluster indicate that some data points are close with some given distance measurement.

2.6.1 Clustering tendency

Before clustering is applied, it is good to know if there is any underlying cluster tendency in the data. If the data for example is uniformly randomly distributed, the clusters will give no insight. In two papers [2], [1] by Adolfsson et al. they test different techniques to determine cluster tendency on numerous data sets. Each data set had been converted to numerical values such that a distance could be measured between instances. They tested both the clusterability and efficiency of the data to determine what technique would be useful.
In their conclusion, certain techniques are more optimal based on the type of data. They performed different tests based on criteria that they believed are useful to consider when testing for cluster tendency. The three criteria were the robustness to outliers, how well the technique performed on overlapping data and if it worked on high dimensional data. The results showed that there was one technique that was most suitable for our data since it was robust to outliers and could handle overlapping and high dimensional data. This technique was the Hartigan's dip test of unimodality by Hartigan & Hartigan [16].

Theory 7: Hartigan's Dip Test of unimodality

Theory 7
First the data is prepared by generating a $n \times n$ matrix of the pair-wise distances of the n data points in the distance space. The test is a statistic that is calculated by taking the maximum difference, over all sample points, between the observed distribution from the distance matrix and a chosen uniform distribution. The uniform distribution is chosen such that the maximum difference between the distributions is minimized (which Hartigan & Hartigan argued to be the best choice for testing unimodality). By repeated sampling from the uniform distribution a sampling observation is obtained. If the statistic is at or greater than the 95th percentile of the sampling observation, the data is at least bimodal. Thus, the statistic is given the null-hypothesis of being unimodal and if $p < 0.05$ the distribution is considered to be at least bimodal. Bimodality, or multimodality yields more value in clustering compared to unimodality.

Based on the paper [5] by Banerjee and Dave, another technique was used to determine the cluster tendency. This technique was the Hopkins statistic [18] by Hopkins & Skellam.

Theory 8: Hopkins statistic

Theory 8

The test is defined by letting X be a set of n data points, sample uniformly at random m of the points into a new data set Y without replacement. This means that the m points are equally probably sampled, such that all features have the same impact on the new data set Y. Based on [5] a good value of points is $m = 0.1 \cdot n$. Let u_i be the distance of $y_i \in Y$ to its nearest neighbour in X and w_i be the distance of $x_i \in X$ to its nearest neighbour in X, then the Hopkins statistic is defined as

$$H = \frac{\sum_{i=1}^{m} u_i^d}{\sum_{i=1}^{m} u_i^d + \sum_{i=1}^{m} w_i^d},$$

where d is the dimension of the data. The closer H is to 0, the more likely it is that the data has an uniform distribution and the data will not have an insightful clustering. The closer it is to 1, the more likely it is that there is a cluster tendency in the data.

To get an indication if the two feature representations have cluster tendencies we applied first the hopkins statistic on both representations and observed that both yielded a similarly high hopkins score. This can be seen in Table 2.1. This means that both methods are viable, but it is not a definitive result to choose either. We also calculated the Hartigan's dip test which showed that Skip-gram had a noticeably better score than CBOW. Therefore, we use Skip-gram for the final pipeline.

Method	Hopkins statistic	Hartigan's dip test
CBOW	0.960163	1.647797
Skip-Gram	0.963907	4.771176

Table 2.1: Table for the investigation of cluster tendency to determine which method is better for Word2Vec. Here the Hopkins statistic has been averaged over 100 iterations and Hartigan's dip test is estimated from the [24] library for HDB-SCAN. The value seen for the dip test is the ratio between the dip statistic and the 95% percentile, which means the larger the value the less of a unimodal distribution of the data. The row marked in yellow is the method that had the highest score for our tests.

2.6.2 Clustering techniques

Once the embedding spaces were generated, we started to consider which clustering technique that was optimal. There are four main techniques of clustering that cluster the data in different ways, hierarchical, centroid-based, graph-based and density-based clustering. Each technique has its own use case and certain assumptions that need to be fulfilled to work optimally. Since we neither knew the shapes nor the number of clusters, we utilized density-based clustering techniques. An additional benefit of using density-based techniques is that the models will identify anomalies in the data and assign them to an anomaly cluster.
A very commonly used algorithm for density-based clustering is DBSCAN [12] by Ester et al. DBSCAN estimates the underlying Probability Density Function (PDF) of the data by transforming the euclidean distance $d(a, b)$ from point a to b by

$$mrd(a, b) = \max\left(c(a), c(b), d(a, b)\right), \tag{2.5}$$

where $c(a)$ is the smallest radii of the hyper-sphere that contains $min\_samples$ neighbours for a. Once the distance is calculated for each point, a minimum spanning tree is built and pruned such that we get a spanning tree for $min\_samples$ neighbours. With a minimum spanning tree, the next step is to convert it into a hierarchy of connected components, which can be seen in Figure C.1 in Appendix C. From the dendogram, the clusters are obtained by drawing a horizontal line at distance ϵ and divide the data into clusters where one of the clusters is an anomaly cluster. The main issue with DBSCAN is that it is sensitive to the choice of the parameter ϵ, which is not very intuitive to set. Also, since ϵ is static, DBSCAN does not allow varying densities. To tackle this, we tried two other algorithms that have been developed with inspiration from DBSCAN, which remove the need to manually set ϵ while improving the efficiency and performance. The first algorithm is OPTICS [3] by Ankerst et al.

Theory 9: OPTICS

Theory 9
OPTICS begins by creating a dendogram in a similar fashion as DBSCAN does, see Figure C.1 in Appendix C, but it utilizes a different mutual reachability distance formula. OPTICS transforms the euclidean distance $d(a, b)$ from point a to b by

$$mrd(a, b) = \max\left(c(a), d(a, b)\right),$$

where $c(a)$ is the smallest radii of the hyper-sphere that contains the $min\_samples$ closest neighbours for a. As the points are processed and the mutual reachability distance is calculated, the algorithm creates a reachability plot, as can be seen in Figure C.3 in Appendix C. From the reachability plot, one can set a threshold for what the minimum reachability distance is to define what points are outliers. Based on the distribution of the reachability distances, the algorithm divides the data into distinct clusters, and the points that have a larger reachability distance than the minimum are assigned as noise.

The benefits of using OPTICS over DBSCAN is that it allows varying densities when deciding the clusters and it has good stability over parameter choices. Also, the choice of OPTICS' ϵ parameter only affects run-time, instead of being an unintuitive and very important parameter for DBSCAN. The design of the algorithm makes it very appealing since there aren't many parameters to set. However, when setting ϵ to its maximum value the runtime goes up noticeably, but this is not a major issue for our pipeline.

The second algorithm is HDBSCAN [6] by Campello et al.

Theory 10: HDBSCAN

Theory 10
HDBSCAN begins by creating a dendogram in a similar fashion as DBSCAN does, see Figure C.1 in Appendix C. To avoid having to select the ϵ in DBSCAN algorithm, another parameter is provided. This parameter min_cluster_size states how many points that are needed to form a cluster. By walking through the hierarchy the algorithm checks at each splitting point if each split has more points than the min_cluster_size. If split a has more and split b has less then a will retain the cluster identity of the parent and b will be marked as 'points fallen out of the cluster parent' and at what distance this happened. If both a and b have more it indicates a true cluster split and let it split there. For a given min_cluster_size yields a condensed cluster tree as can be seen in Figure C.2 in Appendix C. To determine where the clusters are from this plot, calculate the stability

$$s = \sum_{p \in cluster} (\lambda_p - \lambda_{birth})$$

for each cluster, where λ_p is the λ value where the point fell off and λ_{birth} is the λ value when the cluster split off. Afterwards, determine all nodes to be selected clusters. From the bottom up, look at the child clusters and add their stabilities up and compare that to the parent's stability. If the stabilities are larger then the parent gets the sum of their stabilities. Otherwise the parent becomes the selected cluster and the children are unselected. This is continued until the root node is reached, and the selected clusters are returned and the rest of the data is assigned to an anomaly cluster.

The benefits of using HDBSCAN over DBSCAN are similar to the benefits of using OPTICS over DBSCAN.

2.6.3 Clustering evaluation

To determine which clustering algorithm would yield the best clusters for the data, we needed to evaluate the clusters with different metrics. Since we have an unsupervised clustering task, we needed to look at internal evaluation metrics. A popular internal evaluation metric is the Silhouette coefficient [28] by Rousseeuw, Peter.

Theory 11: Silhouette coefficient

Theory 11
Let $a(i)$ be the mean distance between i and all other data points in the same cluster C_i and $b(i)$ be the mean dissimilarity of the same point to all other points in cluster $C_k, k \neq i$

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \tag{2.6}$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j), \tag{2.7}$$

where $d(i,j)$ is the distance between points i and j. Define a Silhouette value

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i), \end{cases}$$

that is limited from -1 to 1, where a value close to one means that the data is appropriately clustered and vice versa if the value is close to negative one. The mean over all Silhouette values is then calculated and returned.

One issue with using the Silhouette coefficient directly on the clusters is that the score will be affected by the anomalies. One idea is to remove the anomalies, which is just to exclude the anomaly cluster from the evaluation. If this is done then the Silhouette coefficient could be used to determine how well each object has been assigned to the remaining clusters. This idea is based on the assumption that the classification of anomalies has been done well. Since we had no specific guidelines on how many anomalies occur, we needed to look for some other metric. One metric that we found is obtained from the Density Based Cluster Validity (DBCV) technique [26] by Moulavi et al.

Theory 12: DBCV

Theory 12
Considering cluster C_i, a Minimum Spanning Tree M_i is constructed from the mutual reachability distance,

$$mrd(a,b) = \max\left(c(a), c(b), d(a,b)\right), \tag{2.8}$$

where $d(a,b)$ is the euclidean distance data point a and b and c is defined as

$$c(o) = \left(\frac{\sum_{i=2}^{n_i}\left(\frac{1}{KNN(o,i)}\right)^d}{n_i - 1}\right)^{\frac{1}{d}}, \tag{2.9}$$

where $KNN(o,i)$ is the distance between object o and its ith closest neighbour, d is the dimension of the data and n are the amount of objects. For each cluster $C_i, 1 \leq i \leq l$, construct l minimum spanning trees and calculate the validity index for cluster C_i

$$V_C(C_i) = \frac{\min\limits_{1 \leq j \leq l, j \neq i}(DSPC(C_i, C_j)) - DSC(C_i))}{\max(\min\limits_{1 \leq j \leq l, j \neq i}(DSPC(C_i, C_j)) - DSC(C_i))},$$

where $DSC(C_i)$ is the maximum edge weight of the internal edges in C_i and $DSPC(C_i, C_j)$ is defined as the minimum reachability distance between the internal nodes of the minimum spanning trees of C_i and C_j.

This technique looks at the entirety of the clusters and produces a score for the quality of the clusters with respect to density. The score goes from -1 to 1 where large values indicate better density-based clustering solutions.

As we do not know the number of anomalies and a desired number of clusters for the data, we felt that we could rely on the DBCV score to find a good clustering. We were first considering using the DBSCAN algorithm, but since we lacked knowledge about the data and selecting its parameters, we started with the HDBSCAN algorithm. This was mainly due to HDBSCAN including a built in estimator for DBCV which allowed us to reduce the number of parameter selections. We experimented with different combinations of the parameters *min_samples*, *min_cluster_size* and observing the DBCV score, number of anomalies and number of clusters. By doing so we limited the possible parameter selections. We noticed that we could not solely rely on the DBCV score since this sometimes gave a large score to a clustering that did not seem to be good, when considering the number of anomalies or the amount of clusters. We ended up with a list of feasible parameter selections, but since the clustering is very important to find the correct anomalies, we investigated if we could reduce the feasible parameters further. For example, feasible parameters would yield a cluster range between 25 to 85 clusters. After consulting experts at Ericsson, we still could not determine a feasible number of clusters, so we looked into algorithms that can analyze the data for us.

One technique that we found was to estimate the number of clusters for a data set using the gap statistic [29] by Tibshirani et al.

Theory 13: Gap statistic

Theory 13
Cluster the observed data, by varying the number of clusters from $k = 1, \ldots, k_{max}$. Suppose there are k clusters $C_1, C_2, \ldots, C_k$, compute the total inter-cluster variation

$$W_k = \sum_{r=1}^{k} \frac{1}{2|C_r|} D_r,$$

where D_r is the sum of pairwise distances for all points inside cluster C_r. Generate B randomly distributed data sets as reference. Cluster the reference data sets by varying the number of clusters from $k = 1, \ldots, k_{max}$ and compute a reference intra-cluster variation W_{kb}. This under the null hypothesis that the clusters have a uniform distribution makes W_{kb} the expected value for W_k. Calculate how much W_k deviates from the expected value by the estimated gap statistic

$$Gap(k) = \frac{1}{B} \sum_{b=1}^{B} \log(W_{kb}) - \log(W_k)$$

and the standard deviation s_k of the statistic. Finally, choose the smallest k such that

$$Gap(k) \geq Gap(k+1) - s_{k+1}$$

where k will yield the clustering structure that deviates as much as possible from an uniform distribution.

After running the gap statistic between 10 and 85 clusters, we obtained the graph shown in Figure C.4 in Appendix C. We noticed that the gap value had almost a monotonic increase, which suggests that the higher the number of clusters is likely to be better for our data. The algorithm suggests that the optimal number of clusters in the data are 85, but this would not be true in our case since we have a big anomaly cluster. We just know that the number of clusters is preferred to be high over low and most likely less than 85. This meant that the gap statistic was not suitable to determine a final parameter selection, since we still had a lot of candidates. To get more information about the clusters we applied the Silhouette coefficient on each parameter selection without the anomalies. We noticed a trend over a lot of iterations that the more clusters we had, the larger the Silhouette coefficient. With all of this information, the final parameter selection that can be seen in Table 2.2, was based on the following information; It yielded a relatively large number of clusters that is less than 85, a reasonable number of anomalies, large DBCV score and a large Silhouette coefficient.

min cluster size	min samples	DBCV score	Silhouette coefficient	Number anomalies	Number clusters
10	30	0.124795	0.808445	2546	90
10	35	0.056367	0.800466	2536	80
10	40	0.067839	0.801046	2766	72
10	45	0.065248	0.789184	2303	55
10	50	0.103754	0.784355	2536	55
10	55	0.079718	0.791009	2537	47
10	60	0.084433	0.795656	2687	45
15	30	0.124422	0.814370	2496	88
15	35	0.055782	0.800510	2546	79
15	40	0.067036	0.799190	2735	71

Table 2.2: Shows a snapshot of 10 parameter selections that were narrowed down during the investigation. The selections were chosen by grid searching on the two parameters *min_cluster_size* and *min_samples* and observing the result in the four other columns. The final parameter selection, which is marked in yellow, was selected by first considering the DBCV value which had a large value compared to other selections. Secondly we could see that the number of clusters were less than 85, which ruled out the rows with a higher DBCV score than the marked row. Also, it did not yield a small Silhouette Coefficient.

We also tested by running OPTICS on the same data set. We initially tested the default parameters but the results were not good since there were a lot of anomalies and large number of clusters. Instead, we tried similar parameters to HDBSCAN to see what kind of result we would get. One issue is that the OPTICS library does not have a built in estimator for the DBCV score and running the full DBCV algorithm for every parameter selection would not be feasible since it is very slow. After running OPTICS for a few parameter selections we noticed that there were way too many anomalies for our liking, since we had obtained a feasible result from HDBSCAN already.

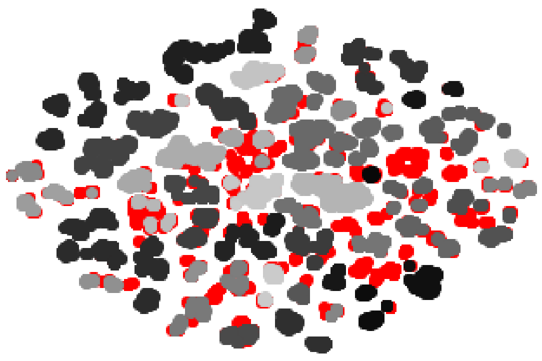


Figure 2.6: 2D representation using TSNE for HDBSCAN for the final parameter selection. Each grey scale color represents a cluster. There are 55 different grey scales in the plot. The largest cluster can be seen in red which corresponds to the 2536 environment anomalies.

Once we had a good parameter selection for HDBSCAN, the next step was to find the configuration setting anomalies inside each cluster except for the environment anomalies, which can be seen in red in Figure 2.7.

2.7 Anomaly detection

Initially, we wanted to utilize anomaly detection methods for each configuration feature for all the environments. Domingues et al. [11] compared 14 different approaches to detect anomalies. We used their work to find candidate methods for the anomaly detection. The result of the study claimed that the Isolation Forest model was the best option since it efficiently identified outliers and was able to find anomalies better in high dimensional data compared to the other methods. The Isolation Forest was introduced in the paper [23] by Liu et al. The Isolation Forest starts by sampling a subset of the given data. By using binary decision trees, it splits the data on the feature values iteratively until a single data point is isolated within the splitting boundaries. The isolation score of a point is the average path length from the root to the point after iterating multiple times. The idea would be that the points found by the Isolation Forest are points that would be removed from the training data. However, we noticed that the anomalies were heavily influenced by the environment parameters, which was not our intention. Hence we wanted to create environment clusters by clustering on only the environment subspace. Each environment cluster would therefore have a low inner variance amongst the environment parameters, except for the anomaly cluster which we would not consider. We did not consider this cluster because there is still much uncertainty about the variance inside, since it contains all points that the model labeled as outliers. For each of the selected environment clusters, we added a single configuration feature to the subspace. This meant that if we now use an anomaly detection method for the data points inside this cluster, most of the variance will be contributed by the added configuration feature, if the clustering is good. The points that are found by the detection are the points that we believed were the ones that should be removed from the training data.

To find the anomalies inside an environment cluster, we considered using Isolation Forest or running HDBSCAN or OPTICS on the data inside to find outliers. However, these algorithms would only consider the data inside the cluster, and not respect the total distribution of the data. As an example, say that a minority class is contained completely inside a cluster. This means that for the given environment cluster, it makes sense for this data to be in here. The anomaly detection models mentioned before will likely label the minority class as an anomaly, since the points are very uncommon inside. Hence, we believed that we needed to create our own anomaly score so that we respect the data's underlying distribution and make decisions whether a data point is an anomaly based on that. For this reason, we ruled out using regular anomaly detection methods. We also could not find any clustering algorithm that respected the data outside the cluster so we empirically created two anomaly detections (method A and method B, that will be defined later) by manually looking into the distributions. Table 2.3 shows an example that represents the distributions we looked at.

cluster frequency	setting frequency	total frequency	configuration setting
0.172414	0.009690	0.213789	4096 qam
0.027586	0.015326	0.021627	2048 qam
0.600000	0.045407	0.158767	4096 qam light
0.075862	0.011305	0.080626	2048 qam light
0.020690	0.012931	0.019224	1024 qam light
0.027586	0.022222	0.014915	1024 qam
0.027586	0.003165	0.104740	256 qam
0.006897	0.004717	0.017567	512 qam light
0.020690	0.030000	0.008286	4 qam strong

Table 2.3: Shows the investigation to determine if any of the specified configuration settings should be classified as an anomaly or not. The configuration settings in the example are found inside a singular cluster. For method B, we consider the cluster frequencies and total frequencies and we can see that for the marked line the cluster frequency is noticeably lower than the total frequency. With the threshold parameter to 0.01 only found this configuration setting as an anomaly. In this cluster there are 145 settings with one belonging to 512 qam light, so this makes sense to be an anomaly.

From this table we could see which configuration settings we considered to be anomalies, based on the cluster frequency, setting frequency and total frequency. The cluster frequency is the frequency of the configuration settings inside the cluster, which sums up to 1. The setting frequency is the frequency of finding a particular configuration setting inside the cluster compared to all other clusters and the total frequency is the frequency of finding a particular configuration setting inside all the clusters. The first anomaly detection method A uses anomaly score A_i that is given by

$$A_i = \frac{\alpha}{n} - \frac{a_i}{||a||}, \quad a_i = \sqrt{s_i}\frac{c_i}{\sqrt{t_i}} \tag{2.10}$$

where s is the setting frequency, i is the ith configuration setting in the cluster, c is the cluster frequency, t is the total frequency, α is a tuning parameter and n is the number of unique configuration settings inside the cluster. By setting α to different values between 0 and 1, we tune the restriction to how far away the point can be from its expected value $1/n$ and therefore control the amount of anomalies we get. The expected value corresponds to whether the values inside the cluster had an equal distribution or not. After running the function for different values and observing the number of anomalies and what kind of anomalies we obtained, 0.5 seemed like a good value to set. From the anomaly function shown in Equation 2.10 we could therefore find which of the values are anomalies and also obtain them in a ranking order from least likely to most likely to be an anomaly. The anomalies are the A_i's that are positive, so the least likely configuration settings that are anomalies are the most negative A_i's.

The second anomaly detection method B uses anomaly score B_i that is given by

$$B_i = \sqrt{n} \cdot \frac{c_i^2}{t_i},$$

where i is the ith configuration setting in the cluster, c is the cluster frequency, t is the total frequency and n is the number of unique configuration settings inside the cluster. To determine if a configuration setting is an anomaly, filter the B_i's by a set threshold. In the final pipeline we found that 0.01 gave a good number of anomalies for the data that we were using. Since there was no good analytical way to determine the best anomaly detection method for our data, we simply chose to use method B in the pipeline.

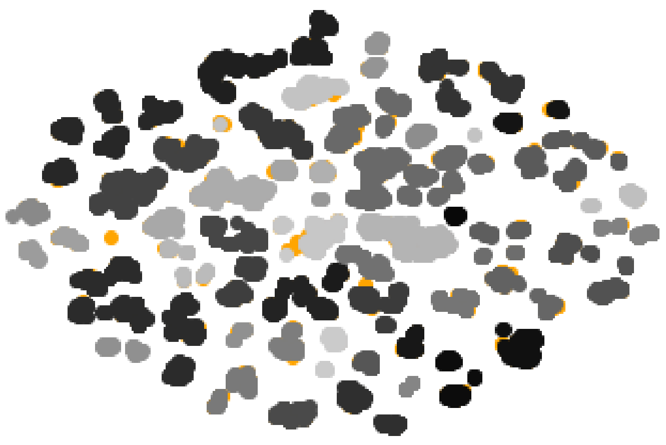


Figure 2.7: 2D representation using TSNE for HDBSCAN for the final parameter selection. Each grey scale color represents a cluster. There are 55 different grey scales in the plot. The environment anomalies we found using HDBSCAN have been removed due to the remaining variance inside that cluster. The cluster which is seen in orange corresponds to the 808 out of the 9532 configuration settings that our anomaly detection method B classified as anomalies.

2.8 Model selection

The predictions of classifiers can be misleading when the data is imbalanced. Classifiers might pick up on the major classes and therefore lead to a good accuracy, even though the classifier is naive and does not pick up on complex relations of the data. As a very first step and a ground baseline for all other investigated classifiers it is therefore a good idea to implement a dummy classifier. As many configuration features have a dominating configuration setting we decided to implement a dummy classifier that always predicts the most frequent class in the training set. This helps to understand whether and how much a smarter and more complex model is contributing to the quality of the predictions. When exploring the possibilities of machine learning in a new field it is usually a good choice to start with tree-based models. Decision tree models are a good first choice for investigating whether the data allows machine learning tasks as they do not require much data, are applied quickly and lead to good predictions in general. As our data set is relatively small and the application of machine learning unexplored this is a first good choice for this project. We use the Random Forest classifier here as a baseline for more complex tree-based models. Random Forest [17] is an ensemble algorithm that was introduced by Ho et al.

Theory 14: Random Forest

Theory 14
Random Forest is an algorithm that was introduced in [17] by Ho in 1995. Random Forest is based upon bootstrap aggregation (bagging) of single decision trees. The bagging is done by taking n random samples of the data, with replacement, and training a decision tree on each subset. Afterwards the majority vote is taken, which prevents overfitting compared to just using a single decision tree for the entirety of the data. However, the same features are used for each subset. This leads to correlation between each sub-model, which is not favorable for the model's prediction. Instead, Random Forest includes sub-sampling of the features as well. This prevents both overfitting and too much correlation between the sub-models.

Another way of creating an ensemble of decision trees is by using boosting, which uses the decision trees sequentially instead of parallel like in bagging. When the decision trees are run in a sequence, it converts the weak learners into stronger ones by attempting to correct the previous learner. One way of accomplishing this is by using Gradient Boosting. Gradient Boosting works on the premise that there is a differentiable loss function that needs to be optimized. It optimizes by iteratively adding weak learners to the model. The weak learners, which are specially constructed decision trees, make predictions and try to minimize the loss function. Without optimization, Gradient Boosting algorithms tend to overfit, due to their greedy nature. There are three recent Gradient Boosting algorithms that have been used frequently in data science competitions that have their own way of increasing performance, speed and reducing overfitting. We cover the main innovations that each model contributed in Theories 15, 16, 17.

The first algorithm is XGBoost [7] by Chen, Tianqi, and Carlos Guestrin.

Theory 15: XGBoost

Theory 15
XGBoost is a Gradient Boosting algorithm that was developed with focus on scalability. For example, XGBoost introduced a novel tree learning algorithm for sparse data and a regularization term to reduce overfitting. When the input is sparse and a missing value is found in the training data, the instance is classified into a default direction. The default direction is how the tree will split at the missing value. For each branch in the trees, there is a default direction, but the optimal direction is learned from algorithm when it runs over the data.
To penalize the complexity of the models XGBoost utilizes both L1 and L2 regularization to prevent overfitting. For example, when XGBoost determines if it should split at a tree node it uses

$$L_{split} = \frac{1}{2}[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}] - \gamma, \tag{2.11}$$

where L and R are the instance sets to the left and right respectively after the split, G and H are the first and second-derivative of the loss function, γ is the minimum loss reduction required to add another partition in the tree and λ is a L2 regularization parameter. These innovations among others is what makes XGBoost a popular choice.

The second algorithm is LightGBM [21] by Ke, Guolin, et al.

Theory 16: LightGBM

Theory 16
LightGBM is a Gradient Boosting algorithm that mainly introduced two new novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS is considering the lack of weighting in regular Gradient Boosting decision trees, and what this does to the information gain. With a given threshold, GOSS down-samples only the data instances that have been over-trained, since these do not contribute as much to the overall information gain. EFB considers a sparse feature space, which it reduces the number of features very effectively on. The reduction is achieved by bundling together exclusive features. Exclusive means that the features rarely take non-zero values simultaneously. The algorithm is then designed to transform the bundling of exclusive features into a graph coloring problem and solving it with a greedy algorithm.

The third algorithm is CatBoost [27] by Prokhorenkova, Liudmila, et al.

Theory 17: CatBoost

Theory 17
CatBoost is a Gradient Boosting algorithm that uses binary oblivious decision trees as base predictors and mainly differs from other Gradient Boosting algorithm because of two features: a specific processing for categorical features and ordered boosting [27]. Oblivious decision trees have the same splitting criterion across each of its levels.
Representing categorical features with high cardinally using one-hot encoding can lead to a great increase on computation time. CatBoost tackles this by first grouping categories using target statistic (TS) and then apply one-hot encoding. TS are numerical values estimating categorical values and Catboost uses a ordered TS, that builds the TS from a history of random permutations of the training data. In the algorithm a new permutation sampled from a set of permutations in every gradient iteration, and is used to create the TS.
Ordered boosting follows the same concept as each step of the Gradient Boosting algorithm is based on the same randomly sampled permutation of the training data. The goal is to receive the residuals for each instance i of n training instances by creating n supporting models. Supporting model M_j is constructed using the first j training instances of the permutation. The residual of instance i is then calculated using the prediction $M_{i-1}(i)$. The residuals and supporting models are then used to create the tree structure. The tree structure is then again used to boost the supporting models. After creating all t tree structures, a left out permutation is used to determine the leaf values of the trees.

To train the models, we had to consider the class imbalance of the data and the selection of parameters.

2.8.1 Class imbalance

Figures 2.4 and 2.5 show a class imbalance in the data. Class imbalance needs to be considered for the training and evaluation of the models. To consider the imbalance, we split the data by considering the distribution of classes, such that both the training and test set have the same class distribution. This is important to not affect the model by introducing a skewed distribution in the test set. This could for example happen if we were to randomly sample the data into test and training. After the split, we first considered oversampling the minority classes in the training data to make the model more robust towards them. Instead we added the class weights as sample weights to respect the distribution. This turned out to not increase the performance in any noticeable way, so we discarded it for the pipeline. For the evaluation of the model, we looked at certain performance metrics that would respect the true distribution and give an insightful result.

2.8.2 Parameter tuning

The scope of this project is bigger than in most machine learning projects as we investigate multiple targets and compare different classifiers in addition. This leaves us with not one but $15 \times 3 = 45$ models and parameter settings to tune. Parameter tuning is usually meant to be a fine tuning of the model and should not have a crucial impact on its performance such as feature selection and engineering do. Therefore, we limited the parameter tuning to a minimum, applying grid search with between 48 and 96 parameter combinations. The models vary quite a bit when it comes to the parameters that are recommended to be considered while tuning. As LightGBM for example uses a leaf-wise growing algorithm for constructing trees this requires different parameters to tune than algorithms with depth-wise growing algorithm. In addition we applied early stopping to prevent the models from overfitting on the training sets. The results showed a slight improvement of prediction performance.

2.8.3 Evaluation of classifier models

As stated earlier we want our models to perform well on non-anomalies and poorly on anomalies.
To measure the prediction performance of our models on the non-anomalies we decided to use two variations of the evaluation metric F1-score: weighted F1 score and macro F1 score. The F1 score [9] was first introduced by Dice, Lee R.

Theory 18: F1 score

Theory 18
For an example binary classification problem with a positive and negative class, there will be four classes that explains how good the classifications was. The four classes are TP (true positive), FP (false positive), TN (true negative), FN (false negative). These four classes can be found in the elements of a generated confusion matrix. From the elements of the confusion matrix one can obtain the precision score which quantifies how precise a model is on the instances that were predicted positive. From the Equations in 2.12 give the definitions of recall (r) *and precision* (p)*.*

$$r = \frac{TP}{TP + FN}, \quad p = \frac{TP}{TP + FP} \tag{2.12}$$

These metrics explains more about the actual predictions than just the accuracy of the classifier. The metrics have their own niche cases when used, but since we were not interested in just one of the metrics, we used the F1 score which is the harmonic mean of recall and precision

$$F1 = 2 * \frac{p * r}{p + r}.$$

In general the F1 score gives a trade-off between precision and recall and is a good choice for evaluation if neither is preferred for the application. The weighted F1 score calculates the score for each label, that is a specific configuration setting, and builds the average F1 score using the number of true instances in the data set as weights. This takes the imbalance of classes into account resulting in a score that gives a good indication for the overall prediction performance of a model. The macro F1 score however does not take the imbalance of classes into account, resulting in an average score that weights the F1 score of every label equally. We can use this score to see if our model is predicting well on all labels, or possibly on the labels that take the major part of the data. In general, we are aiming for good scores in both metrics, but we expect that the macro F1 score will be lower than the weighted F1 score.
For the evaluation of our models on the anomalies we only considered the weighted F1 score. We want to achieve a fairly poor prediction performance on this data set, preferably in absolute terms but at least relatively to the prediction performance on non-anomalies.

2.9 Evaluation of configuration recommendations

The main goal of our application is to increase link performance by changing configuration settings. Therefore, we intended to compare a performance metric for initial configuration settings and the recommendations of our models. A part of such a performance metric was described in Section 2.3. Unfortunately, further performance information was not available during this project. The information available was used to investigate the possibilities to evaluate our recommendations without applying them in reality. As the application of our recommendations in live networks is not possible we rely on the accessible data. However, performance data is only available for real links and their configuration settings and not for the configuration recommendations we provide. Therefore, we tried to approximate the performance values for our recommendations.
One way to approximate the performance metric is to find a fairly similar link for a given environment-configuration pair. We can think of an embedding space that expresses closeness between different environment-configuration pairs in combination with a data structure that enables an efficient lookup for k nearest neighbours. Querying a new environment-configuration pair, in our case the existing link environment with its new configuration settings, will then result in finding k links that are closest to it in terms of its characteristics. As environment and configurations are close we would expect the operational conditions to be similar as well. Given this similarity we can obtain an approximation of the performance of the unknown point by using the performance values of the k nearest links. This idea was implemented using the cat2vec embedding, which is the same embedding used for the clustering and is described in Section 2.4, in combination with a kd-tree as a lookup structure. The kd-tree [13] was introduced by Friedman, Bentley Finkel. In principle a kd-tree is a general form of a binary tree allowing any file to be partitioned in the manner of a binary tree. A partition creates subfiles and also defines lower or upper bounds for the subtrees. These boundaries create multidimensional cells in the search space

and help the nearest neighbour algorithm to ignore instances within cells that are too far away already, and therefore reducing the number of comparisons.
The results of the nearest neighbour search in an early stage of the project were disappointing as the closest neighbour would almost always be the initial link and therefore resulting in a performance difference of 0. This might indicate that changing some configuration settings is not good enough to land on another link. Another possible explanation is that the data set is not big enough as there are no other links close to the combination of environmental features and the recommended configuration settings. For a data set with more links and a higher variance of environment-configuration combinations this approach might work. Either way this approach was not applicable for this project and its data.

Another approach for an evaluation is the discussion of our models' results with experts. This can help to see whether our configuration recommendations are reasonable and in which ways our models can be improved. We were in an exchange with experts from Ericsson in an earlier stage of the project but did not receive enough feedback in a later stage.
This leaves us with no final evaluation of the models for now. A number of iterations including discussions and evaluations with domain experts however seems promising and is something worth looking into in the future.

3
Results and Discussion

The thesis work aimed to explore the possibility to apply machine learning methods to find anomalies in microwave link configuration settings and the recommendations to replace these settings. The available data drew us to the developed approach of identifying possible anomalies using clustering and a simple anomaly score and then excluding anomalies in the training process for models shifting towards the more usual cases of configuration settings. The approach was implemented using the data of two selected customers, as this data seemed the most complete and explored. The data set describes 12068 microwave links including several environmental and configuration features. Through the described process of feature selection and engineering we ended up including 34 environmental features and 14 configuration features, such as modulation, minimum and maximum output power and the bit-error-rate alarm threshold. A complete list of the features and its explanations is provided in Appendix B.
We investigated several possibilities for different aspects of the thesis, such as different cluster algorithms and evaluations for the plausibility of our final recommendations. Investigations with no positive impact were also mentioned in method Section 2 but will not be considered in the presentation of the final results in this section.
Derived from the assumptions about the data we define a good prediction performance for this particular recommendation problem as a precise prediction for non-anomalies and a fairly poor prediction precision for anomalies. Therefore, we aim to increase the weighted F1 score and macro F1 score for non-anomalies and decrease the weighted F1 score for anomalies. However, we consider having a good performance on non-anomalies more important, as we should treat the anomalies resulting from the anomaly detection as anomaly candidates, meaning that a good prediction on a fraction of the anomalies might be desirable as these might not be anomalies after all.

3.1 Evaluating the baseline

Table 3.1 shows the averaged model evaluation over all 14 configurations for each the Random Forest classifier and the dummy classifier when trained on the bigger data set of 12068 microwave links excluding the environmental features from the influx database as this data is only available for a subset of the links. The clustering of environmental features resulted in 55 clusters, and setting the threshold for the anomaly score to 0.01 resulted in 877 anomalies among all configuration features and therefore around 63 anomalies per configuration feature on average.

As already stated the possibilities to improve microwave configuration settings using machine learning have been unexplored until now. Therefore, we investigated the data using a naive dummy classifier that always chooses the most dominant class in the data set. The classes are quite unbalanced in some cases and the prediction performance of more complex classifiers might therefore be misleading when standing alone for themselves, as good results might be achievable when taking advantage of the class distributions.

	F1 weighted	F1 macro	F1 anomalies
Dummy	0.4668	0.1443	0.2954
Random Forest	0.8967	0.6938	0.1865

Table 3.1: Comparing evaluation metrics of baseline models. The highlighted cells show the best results of the according evaluation metric according to our definition of a good performance for our models.

The results for the dummy classifier show that even though there are configuration features with very dominant classes the average prediction performance is quite poor.
We then trained the Random Forest classifier on the same data set. The Random Forest classifier clearly outperforms the dummy classifier as it shows a weighted F1 score about twice as high and a macro F1 score almost 5 times as high. This shows that the data indeed allows the application of machine learning algorithms as the Random Forest classifier picks up meaningful relations within the data. The macro F1 score is however still quite low compared to the weighted F1 score, indicating that the model struggles with minority classes. This could mean that the model misses more complex relations or simply lacks necessary features to explain the remaining variance of the targets.
The predictions on the anomaly data set show that anomaly links, according to the anomaly detection, indeed differ from other links and the Random Forest classifier is not able to predict the right configuration settings with the relations it picked up on the normal links. This might also be due to the generally lower frequencies of configuration settings appearing in anomaly links.

3.2 Comparing gradient boosting classifiers

As a next step we considered more complex models with the goal to push prediction performance further. We decided to investigate the potential of different implementations of gradient boosting algorithms, namely XgBoost, LightGBM and CatBoost. We used the same data set as for the baseline models. The results of the evaluation are stated in Table 3.2.

All gradient boosting algorithms are able to outperform the Random Forest Classifier as weighted F1 score and macro F1 score increase slightly. Contrary to our

	F1 weighted	F1 macro	F1 anomalies
XGBoost	0.9139	0.7476	0.3087
LightGBM	0.9111	0.7376	0.2513
CatBoost	0.8987	0.6906	0.2205

Table 3.2: Comparing evaluation metrics of different gradient boosting models. The highlighted cells show the best results of the according evaluation metric according to our definition of a good performance for our models.

definition of a good model performance, the F1 score for the anomaly data set increases as well. As already stated we focus on a good prediction performance on the non-anomalies and should evaluate the prediction performance on anomalies with care.
Even though LightGBM and CatBoost include a special feature representation for categorical data, as stated in Theory 17, the algorithms are not able to outperform XGBoost. The results are close and only differ in a small magnitude, which is not surprising as the the core idea of all algorithms is the same and an improvement of performance becomes harder when getting closer to 100%. XGBoost seems to work best for our application and we therefore proceeded to investigate possible improvements of this model.

3.3 Recommendation results from XGBoost

Table 3.3 shows the top 10 recommendations resulting from the XGBoost model that was evaluated in this step and is sorted by the anomaly scores. Regarding our considerations about the anomaly score, a lower score indicates a lower likelihood for the configuration setting to be right. However, it should not be interpreted as an absolute probability but rather be considered in relative terms. Such a list of recommendations represents the final result of our application. The table shows different configuration setting changes for specific microwave links. Their true link ids were replaced by consecutive numbers as we are not allowed to publish provider-specific information.
Evaluating the value changes is difficult as incorrect configuration settings are not labeled, identifying true wrong configuration settings in a different way is very difficult and other possible options as described in Section 2.9 failed because of missing data and participation of domain experts. The right choice of configuration settings is influenced by many factors, such as the environmental features, which we had access to and integrated in the model, and other information such as restrictions resulting from a network planning process or a provider in general, e.g. the frequencies a provider has the license for. The model might pick up some of this information implicitly, but the results from Table 3.2 already show that there seems to be an information gap as some configuration settings are mispredicted. We further investigated this by considering additional features for the models. The results are shown

in the following section.

anomaly score	id	configuration	old	new
0.000012	1	modulation	512 qam	4096 qam
0.000039	2	input_alarm_threshold	-56.7	-92.7
0.000045	3	modulation	512 qam	4096 qam
0.000123	4	atpc_max_output_power	17.33	25.33
0.000128	5	max_output_power	25.5	23.0
0.000128	6	modulation	4 qam	4096 qam
0.000150	7	input_alarm_threshold	-56.7	-92.7
0.000150	8	input_alarm_threshold	-56.7	-92.7
0.000150	9	input_alarm_threshold	-56.7	-92.7
0.0001504	10	input_alarm_threshold	-56.7	-92.7

Table 3.3: Top 10 recommendations taken from the XGBoost Classifier trained on the big data set. Old values show the configuration settings present in the data and new values show the recommendations resulting from our models. E.g. recommendations for links 5,6 and 7 suggest a change to a higher QAM modulation, which allows a higher capacity for the link but also increases the chance for errors during transmission.

3.4 Evaluating the impact of additional environmental features

The previous section of the results are based on models that only consider environmental features from the impala data base. As a next step we tried to make use of a) features from a time series database of an operation condition classifier and b) of possible dependencies among configuration features. A list of all features, their explanations and where they come from is provided in Appendix B. As already stated in Section 2.1.1 the scope of the classification application is limited to a subset of the links. We obtained classification data for 6729 out of 12068 links. The new features provide information about what influences a link's operation condition and additional information about the location of links. The results for training XGBoost classifiers on the new data set are presented in Table 3.4, again showing the average scores among the classifiers for all configuration features. The first row shows the results for training the classifiers on the subset without the additional features. This is necessary to observe the impact of additional features as the subset of links in this data set is specific as a majority of these links are placed in urban areas.

The second row shows the results of XGBoost classifiers trained on the same data set including the additional features from the Influx database. In the last row, we included all other configuration features as environmental features in addition to the features from the influx database. A detailed list of evaluation results when using these additional features can be seen in Table D.3 in the Appendix D.

	F1 weighted	F1 macro	F1 anomalies
	0.9293	0.6961	0.3948
+ influx	0.948	0.7544	0.3501
+ configurations	0.9745	0.8733	0.5384

Table 3.4: Comparing the averages of evaluation metrics for a subset of the data set. In two steps we added features from an operation condition classification and remaining configuration features as additional environmental features.

Adding features from an operation condition classifier

Adding the features from the influx database results in 30 clusters and a total of 338 anomalies, therefore around 24 for each configuration feature on average. The additional features push our models a little bit further as weighted F1 score and macro F1 score increased. To what extent these features have an impact can be investigated using feature importance plots. All feature importance plots are given in Appendix E. An example is provided by Figure 3.1. We used the feature importance score *gain*, indicating the contribution of a feature on decisions in the decision trees relatively to all other features. When looking at the lower F1 score for the anomalies we have to keep in mind that the new features are also influencing the clustering and therefore the anomaly detection.

It is not surprising that features like the hop length have an impact on the model performances, as this is information that is also considered when configuring microwave links in the first place. However, information about the operation condition is unknown at that point but still shows a relation to the configuration settings. This means that a link whose operation condition is affected in a certain way ,e.g. by wind, tends to be configured differently. As the model picks up a relation to meaningful configuration settings for different operation conditions, configuration recommendations might become more meaningful. Again, this is something we cannot evaluate at this point.

Considering dependencies between configurations

We went one step further and considered possible dependencies between configuration settings. This means that a setting for one configuration feature might affect

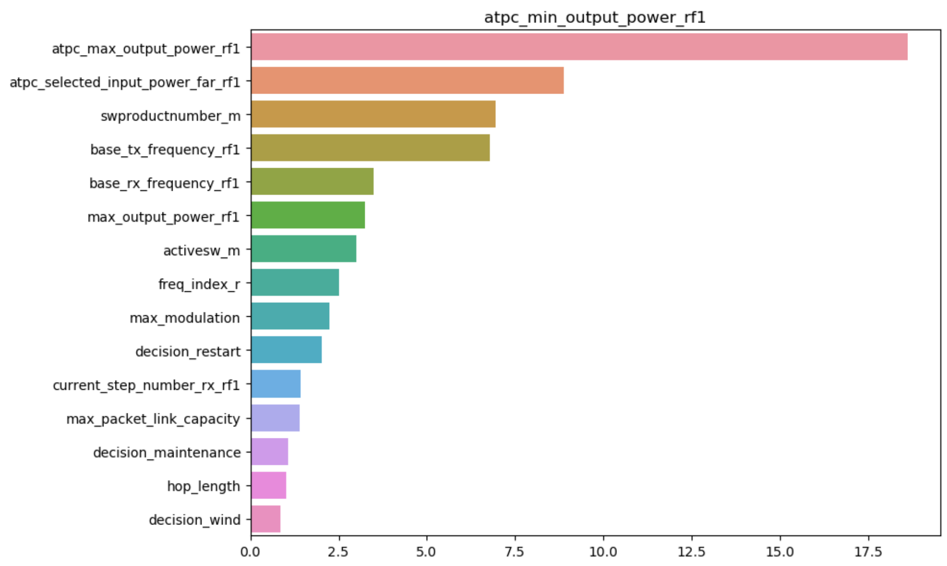


Figure 3.1: Feature importance plot for the configuration feature *min output power for automatic transmission power control*. This plot gives an example of how the additional features from the influx database have an impact on the decision making. Such additional features are: *decision_restart*, *decision_maintenance*, *decision_obstruction* and *hop_length*. The feature importance values speak in relative terms, expressing a feature's influence on decisions in decision trees compared to all other features.

the configuration setting of a different configuration feature. We already explored direct correlations between configuration settings using Cramer's V and association matrices in Section 2.2 and removed configuration features from our scope that seemed redundant. An association matrix for the configuration features is given in Appendix A.2. The remaining configuration features show rather low associations between each other, but there might be more complex relations between them. A combination of surrounding configuration settings might then help to predict the correct configuration setting for the configuration feature the model is trying to predict. Therefore we included the surrounding configuration features when training the models. However we did not consider these features for the clustering.
Including other configuration features as environmental features pushes our models even further as weighted F1 score and specially macro F1 score increase one more time by around 0.025 and 0.086. Other configurations in fact seem to have a quite big impact on some of the configuration features as the prediction performance for some configuration features increases. This can be seen in Table D.3 in the Appendix. E.g. the models for the configuration features *max_output_power* and *channel_spacing* do not improve much when adding the configuration features to the context of a link whereas the weighted F1 score for the model of feature *modulation* improves by another 0.12.

The results from Table 3.4 and the feature importance plots from Appendix E show that some configuration features are quite dependent on other configuration features.

anomaly score	id	conf	old	new
0.000141	11	ber_alarm_threshold	10eMinus3	10eMinus6
0.000247	12	max_output_power	17.0	21.0
0.000286	6	modulation	4 qam	4096 qam
0.000297	4	atpc_max_output_power	17.333333	25.333333
0.000309	13	modulation	512 qam	2048 qam light
0.000434	14	max_output_power	19.0	23.0
0.000461	15	ber_alarm_threshold	10eMinus3	10eMinus6
0.000461	16	ber_alarm_threshold	10eMinus3	10eMinus6
0.000461	17	ber_alarm_threshold	10eMinus3	10eMinus6
0.000581	18	max_output_power	19.0	20.0

Table 3.5: Top 10 recommendations taken from the XGBoost Classifier trained on the smaller data set with influx data and configuration features as environment

We therefore conclude that respecting these dependencies between configuration features is crucial for a recommendation task as of this thesis work. The final models show fairly good prediction performance on the normal links. The fairly good prediction performance of 0.5534 on the anomalies indicates, that there are possibly far less misconfigurations in the data set then we expected. We could therefore consider to lower the threshold of the anomaly detection.
We finally have a look at the top 10 recommendations of these models in Table 3.5. We can see that recommendations for id 1 and 9 are the same as from the models without any additional features. Again, we cannot say much about the reasonability and potential performance improvement of our recommendations as we lack a way to evaluate them.

4
Conclusion

In this thesis we investigated the possibilities to configure microwave links with a data-driven approach. We were specially interested in identifying incorrect configurations and providing suitable configuration recommendations for these cases. We assumed that the number of incorrect configuration settings is low for the network we looked into. Incorrect configuration settings would then appear rarely and could be identify as anomalies. We implemented a machine learning approach that first identifies anomaly candidates for a certain configuration feature, secondly trains a model excluding these anomaly candidates and at last provides recommendations for the anomaly candidates. This resulted in an anomaly detection and classifier model for each configuration feature in scope.
To locate the anomalies we first tried applying anomaly detection methods on the environmental features and the chosen configuration setting. This did not work as a lot of variance came from the environmental features. Instead, we used the density-based clustering algorithm HDBSCAN to cluster the environmental features. As the clustering of microwave links according to their environmental features is an unsupervised problem, it remained difficult to tune and evaluate the clustering until the end. The final clustering was obtained from observing DBCV score, Silhouette Coefficient and the number of clusters and anomalies. These four values helped to find a clustering that appears to fit the data.
Microwave links within the resulting clusters were similar in their environmental features. We empirically developed two anomaly detection scores to point out atypical configuration settings for each cluster. The scores are mainly based on frequencies of configuration settings within a cluster and within the total of all microwave links. Since there was no way to evaluate the anomaly detections we simply chose one of them.

The results show that machine learning models are a viable option to help find suitable configurations for microwave links. The first data set consisted of information for 12068 microwave links and we excluded about 63 anomalies on average among all configuration features. Environmental features consisted mostly of hardware and software information. The models picked up relevant relationships and were able to give fairly good results overall. We trained different Gradient Boosting algorithms on this data set. XGBoost outperformed LightGBM and CatBoost and we continued to use XGBoost in the next steps.
The prediction for less frequent configuration settings seemed more difficult for the models in most cases. We tried to improve the models adding additional environmental features. The prediction performance increased when adding additional

location information and averaged information about operation conditions. In a next step we considered dependencies between configuration features and therefore used other configuration features as environmental features. The prediction performance increased once more, proving that there are dependencies between configuration settings. The final set of models had an average weighted F1 score of 0.9744 and and average macro weighted F1 score of 0.8692. We believe that there is more information respected in the original configuration process. Such information could push our models once more, leading to a better prediction and a better reliability when detecting anomalies.

Applying the trained models on the sets of anomaly candidates showed a lower prediction performance. This was excepted as we desired our models to predict good or common configuration settings and not uncommon settings that might be incorrect. We interpreted the mispredictions of our models as configuration recommendations. The evaluation of these results is difficult and we are left with a lot of uncertainty for the reliability of the anomaly detection and the quality of our configuration recommendations. This is due to the unsupervised nature of the problem as we do not know which configurations are in fact incorrect and indirect ways to check this, e.g. evaluating the performance of microwave links that might be configured incorrectly, were not possible in the scope of this thesis work.
Our approach shows that data-driven configuration for microwave networks and the recommendation of more suitable configuration settings is in principle possible. This is first investigation of the problem and configuration predictions of our models need to be shifted into the right direction and be restricted by the boundaries of a network's reality. This requires additional meaningful features and a reliable evaluation process for the anomaly detection and configuration recommendations.

5 Future Work

At the end of this thesis we are left with many interesting opportunities to build on this thesis work. Some of them acknowledge the limitations of the project and try to break them, others show different approaches for further investigations of the problem. The main ideas for these opportunities are stated in the following sections.

5.1 Utilization of performance data

The main point that needs to be investigated is the evaluation of the steps in the pipeline. Our initial idea was to create a performance function that would describe the overall performance for a microwave link over a given interval. Since we were not able to construct the function, further investigation would be necessary. We could construct such a function for a bigger data set and richer performance data. The design of the function could be based on evaluation from visualization tools to understand how the parameters changes the performance score. With a performance function certain steps in the pipeline could have been quite different.

There could have been an evaluation for the anomaly detection. If we assume that misconfigurations have a less than average performance score over a long interval, then this should be reflected in the identified anomalies. For example, if we take the average performance scores of the anomalous and normal microwave links, we would expect the average score of the normal links to be larger. Therefore, a good evaluation of the anomaly detection is the difference between the two averages, where a large difference indicates that the method has identified anomalies well. With this evaluation we could both tune the anomaly detections and selected a superior method from our two candidates, instead of simply selecting one. Also, with a performance function we could have added weights for the models based on the performance scores. This could have been a great addition to the prediction by placing emphasis on microwave links with a low performance score.

Finally, we could have properly evaluated the result of the pipeline. Once we get a recommendation for a configuration setting we can see the difference between the old performance value and the estimated performance value from similar microwave links. Similar microwave links can be found with nearest neighbour algorithms for the configuration and environment space that all the microwave links span. The performance difference makes ranking the recommendations possible in terms of the largest performance increase. The ranking of recommendations based on performance difference is more insightful than ranking based on the most anomalous, because the performance score would be based on live data. With the highest ranked recommendations, these would be the most interesting ones to analyze further.

5.2 Investigating different networks

Our results are based on the data of a network who's configuration settings have been quite optimized already. It would be interesting to see how our approach performs on the data of network that has not been optimized. This brings up the opportunity for a knowledge transfer, as we could train our models on optimized networks and provide recommendations for non-optimized networks.

5.3 Investigating more data sources

The selection of meaningful features took a large part of this thesis already. However, our feature selection does not explain all of the configuration settings' variance yet. We suspect that the configuration process is guided by additional data, such as certain boundaries for power levels and frequencies. Such additional planning information could help to make predictions of our models more reliable and therefore identify anomalies with more certainty. The usage of the available data sources for this project were just a first step and utilizing more meaningful data sources can push our models further.

5.4 Applying constraints on the recommendations

Currently, the recommendations do not have any rules to the values it recommends. For the recommendations to be insightful they need to be constrained within ranges of acceptable values. The constraints could be applied in two different places, filtering on the recommendations and applying weights to configuration settings. The first constraint could be applied on the final list of recommendations, by creating a rule-based system that filters out all values that can not be configured. The latter constraint could be applied by investigating if the model needs a bias towards a certain subset of the configuration setting's value range, such that it rarely recommends the values outside of the subset. These constraints would be very useful if the recommendations should be considered for production, so that no false settings are applied.

5.5 Wrapper approach

The three gradient boosting algorithms that we used find different kinds of patterns in the data because of their differing architecture. Since all methods did well on the data, it might be worth investigating what a wrapper around the three classifiers would do. Then, a majority vote could be taken from their respective output and we would see how that would affect the predictions.

5.6 Neural Network Approach

Since we were investigating the feasibility of utilizing machine learning approaches on the unexplored data, we wanted the models to be simple. Based on the result we know that machine learning is viable and so the question is how would the result change if a more complex model were used instead. One idea that we had was to use a neural network instead of decision trees. Neural networks are good at capturing the underlying relationship between features and targets, and since there is still room for improvement based on the results, this could be interesting to look into. Additionally, if a performance function were given, we would be able to find an optimal configuration by redefining the problem as:

For a neural network approach we want to approximate a performance surface over all possible context vectors $\bar{y}$ and configuration vectors $\bar{x}$ that is based on the values of the performance function. First, we would train a neural network by using the vector $[\bar{x}_0, \bar{y}_0]$ as input and the respective performance value as output. Once the model is trained, to find the optimal configuration is to run the network on every combination of $[\bar{x}, \bar{y} = \bar{y}_0]$ and assign the configuration vector that corresponds to the highest value that the network will output. This could be feasible if the number of permutations of the configuration vector is not too large. The formulation of this approach was inspired by the work of Haigh et al. [15]. They used a neural network to approximate a performance surface for devices in mobile ad hoc networks based on configurations and environmental information.

The two main reasons to why this approach was not used was that we wanted a simple machine learning algorithm to explore the data set, and that considering the data size we would not be able to design a deep network since there was not much data to train on.

Bibliography

[1] Andreas Adolfsson, Margareta Ackerman, and Naomi C Brownstein. To cluster, or not to cluster: How to answer the estion. *TKDD '17, Halifax, Nova Scotia, Canada ACM*, 2016.

[2] Andreas Adolfsson, Margareta Ackerman, and Naomi C Brownstein. To cluster, or not to cluster: An analysis of clusterability methods. *Pattern Recognition*, 88:13–26, 2019.

[3] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[4] A. Awada, B. Wegmann, I. Viering, and A. Klein. Optimizing the radio network parameters of the long term evolution system using taguchi's method. *IEEE Transactions on Vehicular Technology*, 60(8):3825–3839, Oct 2011.

[5] Amit Banerjee and Rajesh N Dave. Validating clusters using the hopkins statistic. In *2004 IEEE International conference on fuzzy systems (IEEE Cat. No. 04CH37542)*, volume 1, pages 149–153. IEEE, 2004.

[6] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[7] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[8] Harald Cramér. *Mathematical methods of statistics*, volume 43. Princeton university press, 1999.

[9] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[10] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29, 2004.

[11] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74:406–421, 2018.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[13] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977.

[14] David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. A closer look at skip-gram modelling. In *LREC*, pages 1222–1225, 2006.

[15] K. Z. Haigh, S. Varadarajan, and Choon Yik Tang. Automatic learning-based manet cross-layer parameter configuration. In *26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06)*, pages 84–84, July 2006.

[16] John A Hartigan, Pamela M Hartigan, et al. The dip test of unimodality. *The annals of Statistics*, 13(1):70–84, 1985.

[17] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[18] Brian Hopkins and John Gordon Skellam. A new method for determining the type of distribution of plant individuals. *Annals of Botany*, 18(2):213–227, 1954.

[19] H. Hu, J. Zhang, X. Zheng, Y. Yang, and P. Wu. Self-configuration and self-optimization for lte networks. *IEEE Communications Magazine*, 48(2):94–100, 2010.

[20] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications*, 24(2):98–105, 2017.

[21] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., 2017.

[22] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima. Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks. *IEEE Access*, 6:32328–32338, 2018.

[23] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[24] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar 2017.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[26] Davoud Moulavi, Pablo A Jaskowiak, Ricardo JGB Campello, Arthur Zimek, and Jörg Sander. Density-based clustering validation. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 839–847. SIAM, 2014.

[27] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6638–6648. Curran Associates, Inc., 2018.

[28] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[29] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[30] U. Turke and M. Koonert. Advanced site configuration techniques for automatic umts radio network design. In *2005 IEEE 61st Vehicular Technology Conference*, volume 3, pages 1960–1964 Vol. 3, May 2005.

[31] M. N. ul Islam, R. Abou-Jaoude, C. Hartmann, and A. Mitschele-Thiel. Self-optimization of antenna tilt and pilot power for dedicated channels. In *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 196–203, 2010.

[32] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[33] Ying Wen, Jun Wang, Tianyao Chen, and Weinan Zhang. Cat2vec: Learning distributed representation of multi-field categorical data. 2016.

A

Appendix 1: Association matrices

Using the Cramer's V we calculate and plot an association matrix for all possible candidates of environmental features. The matrix is similar to a correlation matrix, where a higher number, or warmer color, indicates an association between two features, meaning that the values of these features are more likely to come in pairs.

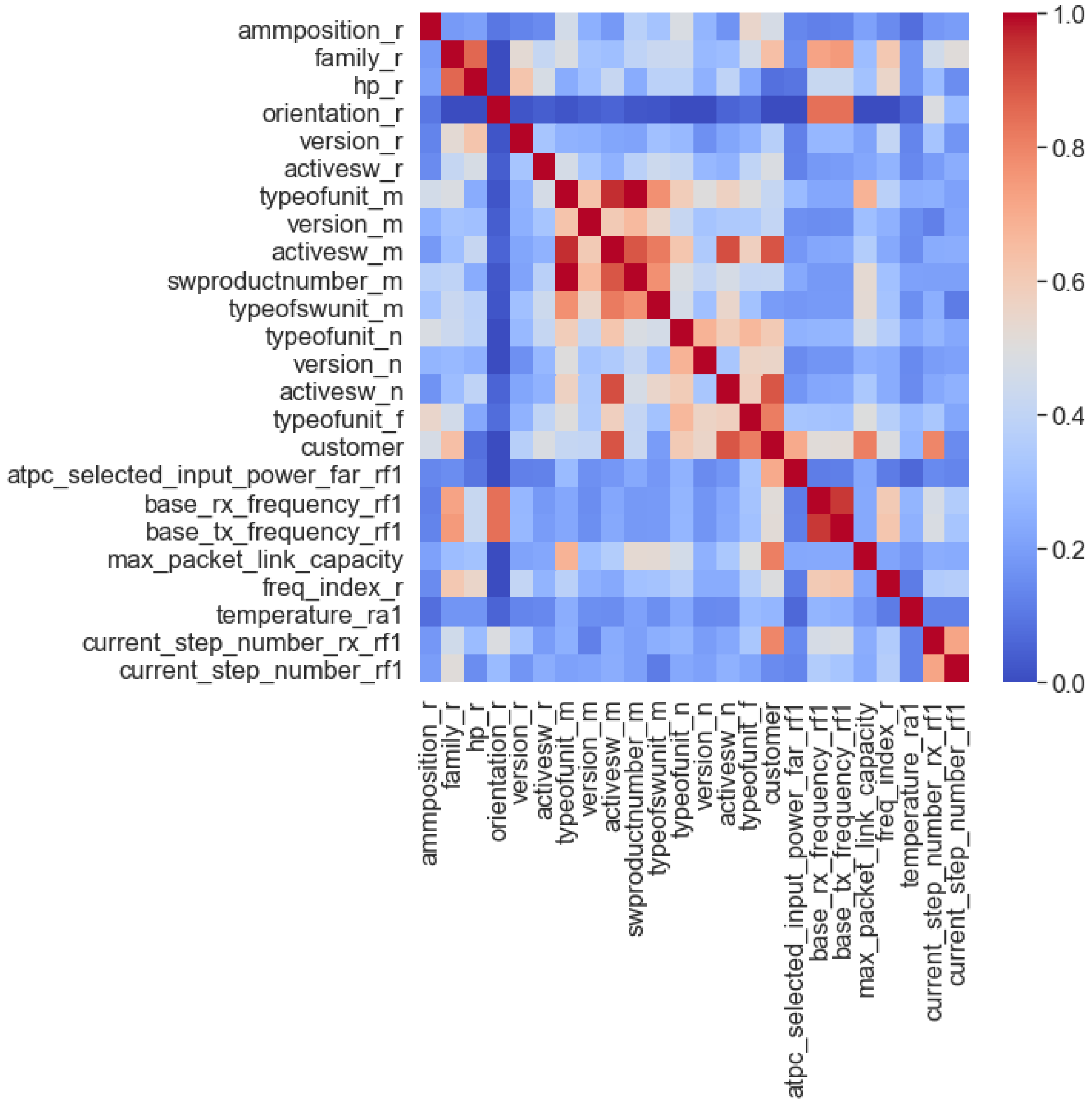


Figure A.1: Heatmap showing associations between environmental features.

The same goes for the pre-selection of configuration candidates.

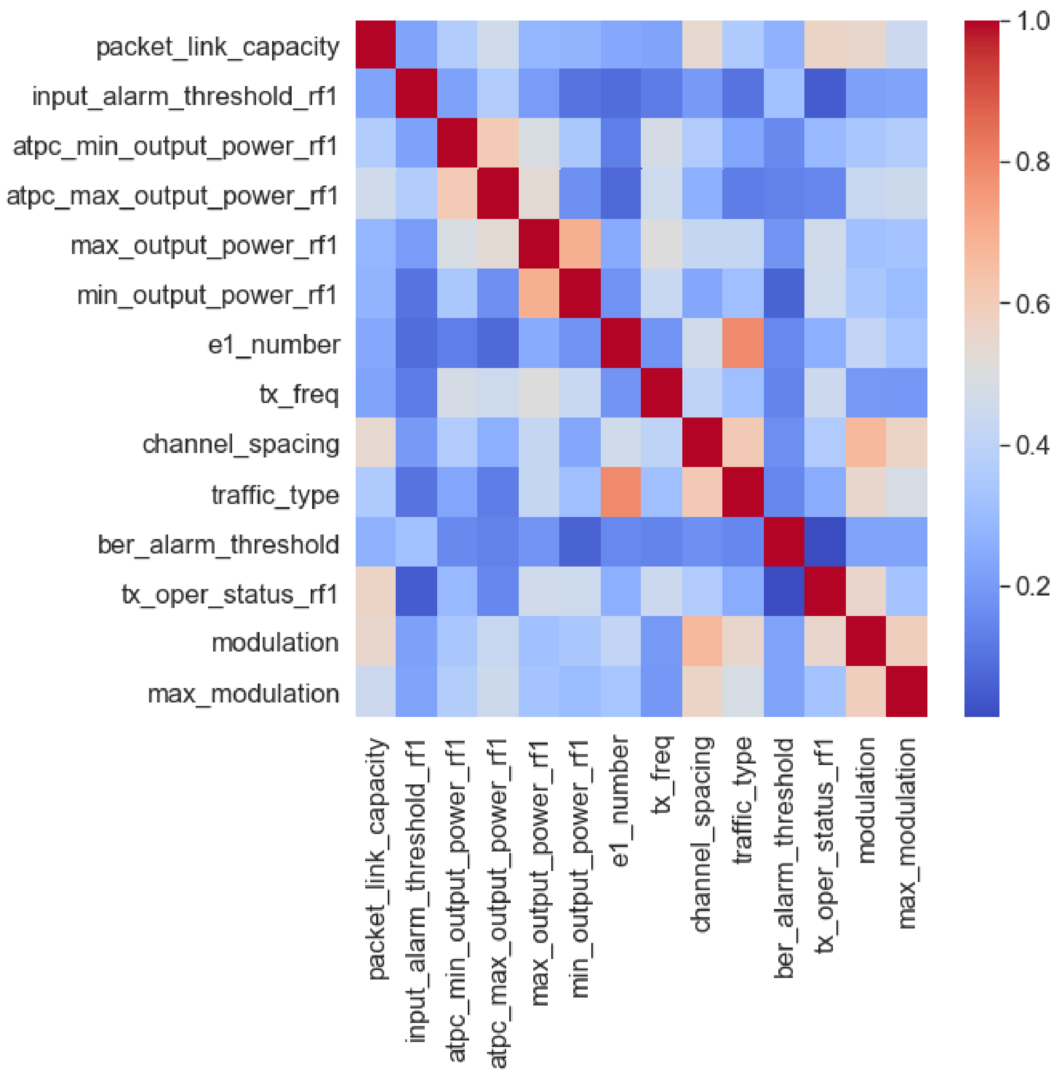


Figure A.2: Heatmap showing associations between configuration features.

The association matrix between environmental features and configuration visualizes how certain environmental features explain variance in some configurations directly.

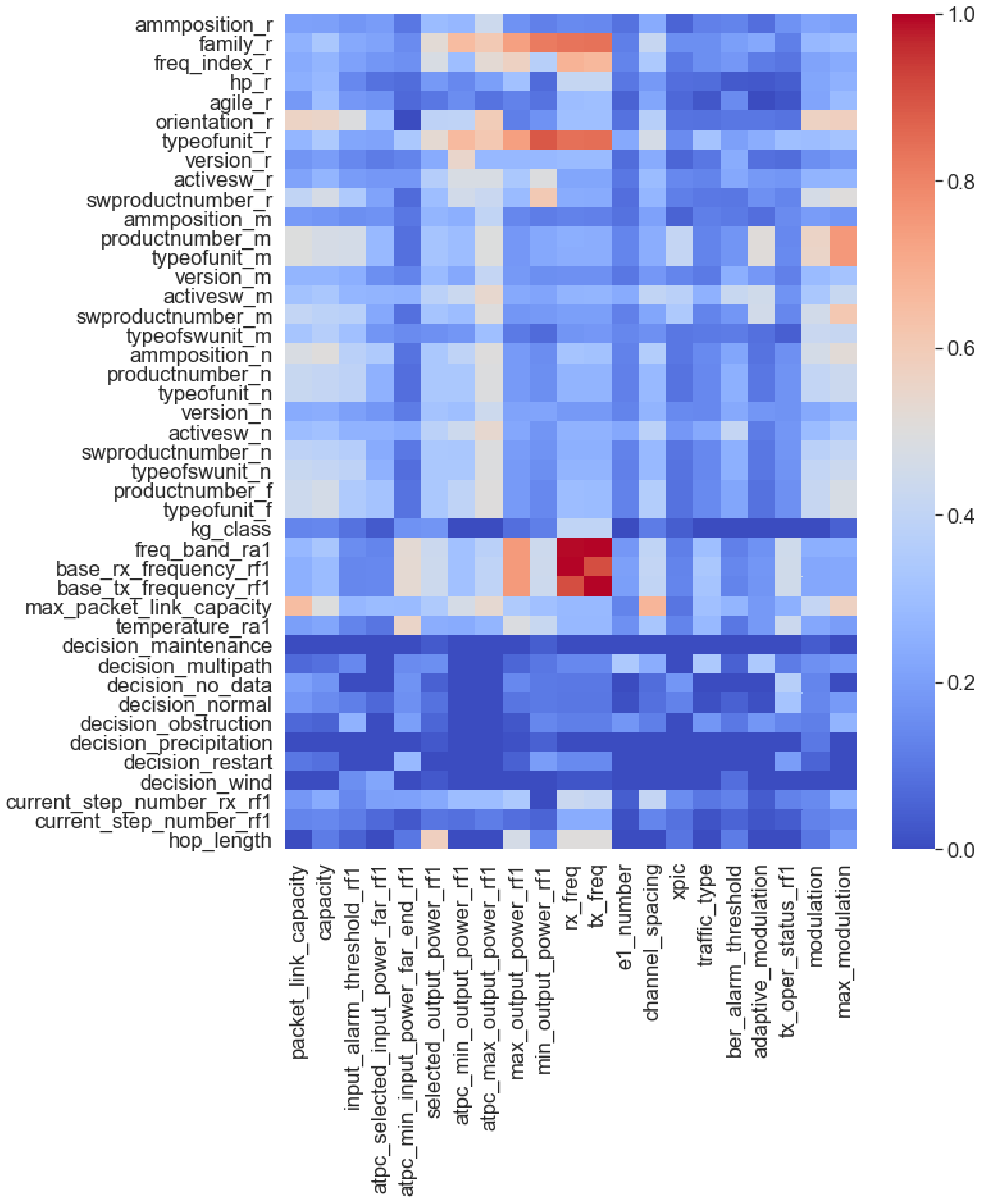


Figure A.3: Heatmap showing associations between pairs of environmental and configuration features.

B Appendix 2: Feature explanations

Table B.1 lists all considered environmental features and a short explanation of their names.

feature name	**explanation**
ammposition_r	Physical position of the radio within the rack
family_r	Name for radio family
hp_r	Flag for high power radio
orientation_r	Indicates whether the radio is operating on the lower or higher part of the radio family's frequency.
version_r	Version of the radio
activesw_r	Software used for the active operations of the radio
typeofunit_m	Type of modem unit
version_m	Version of the modem
activesw_m	Software used for the active operations of the modem
swproductnumber_m	Product number of the software used for the modem
typeofswunit_m	Type off the software unit for the modem
typeofunit_n	Type of unit of the node processor unit
version_n	Version of the node processor unit
activesw_n	Software used for the active operations of the node processor unit
typeofunit_f	Type of unit of the fan unit
kg_class	Climate classification according to Köppen-Geiger. Is based on location of a links near end.
customer	Name of the customer. Some configurations seem to be customer-specific.

atpc_selected_input_power_far	Desired input power for the receiver of the far end node this link. The Automatic Transmit Power Control will try to adjust the transmit power to achieve this input power.
base_rx_frequency	Frequency for down-conversion in the receiver of the link.
base_tx_frequency	Frequency of transmit carrier of the link
max_packet_link_capacity	Maximum for the packet link capacity of a link
freq_index_r	Index for frequency sub-band
temperature_ra1	-
decision_maintenance	Percentage of time the link's signal has been classified as being under by maintenance
decision_multipath	Percentage of time the link's signal has been classified as being affected by multipath
decision_no_data	Percentage of time no classification data was available
decision_normal	Percentage of time the link's signal was was classified as normal
decision_obstruction	Percentage of time the link's signal has been classified as being affected by obstruction
decision_precipitation	Percentage of time the link's signal has been classified as being affected by precipitation
decision_restart	Percentage of time the link's signal has been classified as being restarted
decision_wind	Percentage of time the link's signal has been classified as being affected by wind
current_step_number_rx	-
current_step_number	-
hop_length	Distance between near and far end of the link measured in kilometers

Table B.1: List of explanations for environmental features

Table B.2 lists all considered configuration features and a short explanation of their names.

feature name	**explanation**
packet_link_capacity	Current packet link capacity

input_alarm_threshold	An alarm goes off when the input power goes below the threshold level
atpc_min_output_power	Minimum output power for the Automatic Transmit Power Control (atpc). Is measured in decibels (dB).
atpc_max_output_power	Maximum output power for the Automatic Transmit Power Control (atpc). Is measured in decibels (dB).
max_output_power	Maximum output power of the microwave transmitter. If the maximum output power is bigger than the minimum output power the Automatic Transmit Power Control (atpc) is activated. Is measured in decibels (dB).
min_output_power	Minimum output power of the microwave transmitter. If the minimum output power is smaller than the maximum output power the Automatic Transmit Power Control (atpc) is activated. Is measured in decibels (dB).
e1_number	-
tx_freq	Frequency for the transmitter of the microwave node. IS measured in Herz.
channel_spacing	Bandwidth the signal is allowed to occupy
traffic_type	Traffic type, like ETH or PDH
ber_alarm_threshold	An alarm goes off when the bit-error-rate exceeds the threshold level
tx_oper_status	-
modulation	modulation method for information transmission, like qam 4 or qam 512
max_modulation	When adaptive modulation is enabled this describes the maximum for the adaption

Table B.2: List of explanations for configuration features

C

Appendix 3: Clustering

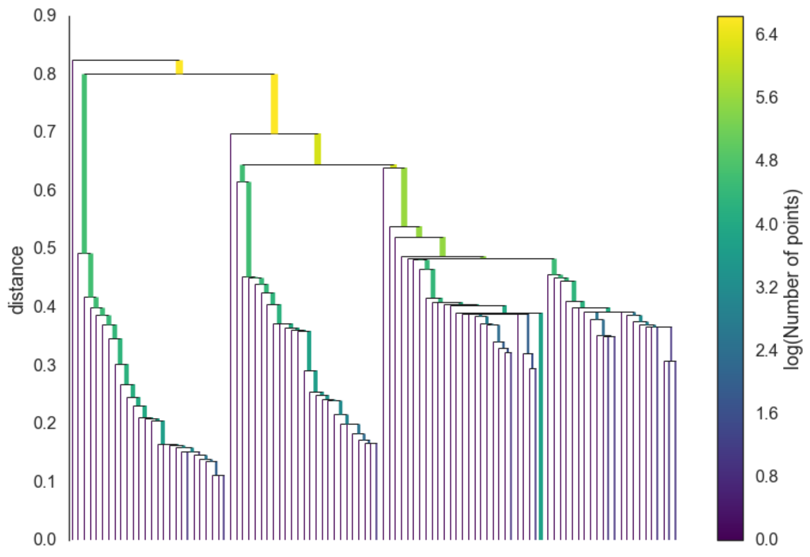


Figure C.1: Plot of a dendogram that illustrates the hierarchy of connected components. The y-axis consists of the mutual reachability distance and the color corresponds to the log of the number of points. To divide the data into clusters, a parameter ϵ is needed for the algorithm. With a given ϵ a horizontal line is drawn in the plot and all the hierarchies below the line are divided into clusters and the data above the line is returned as a noise cluster

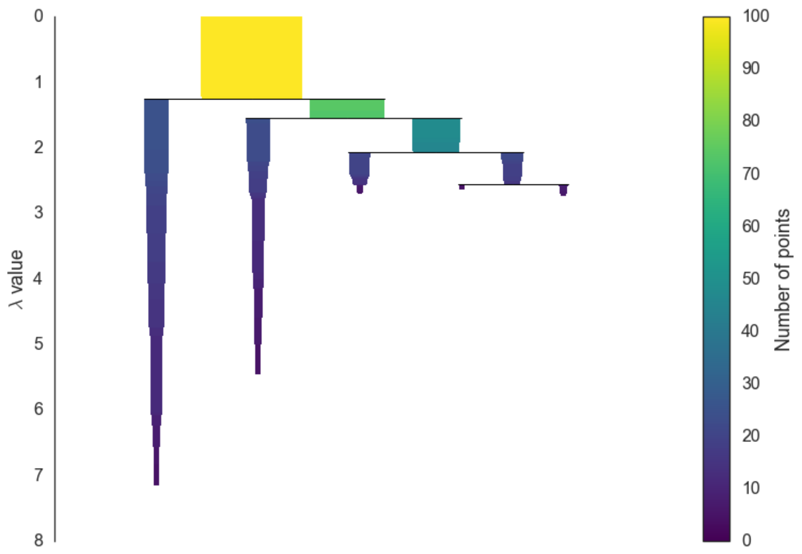


Figure C.2: Plot of a condensed cluster tree where the colors correspond to the amount of points and the y-axis represents the λ value which is defined as $\lambda = \frac{1}{d}$, where d is the mutual reachability distance. For this plot, one can see that there was three times when the two splits had more points than a given *min_cluster_size*, which corresponds to the three rectangular shapes on top of the lines. There was also one time when the splits were both lower than *min_cluster_size*, which can be seen in the final split to the right, where there was two small blue shapes under the line

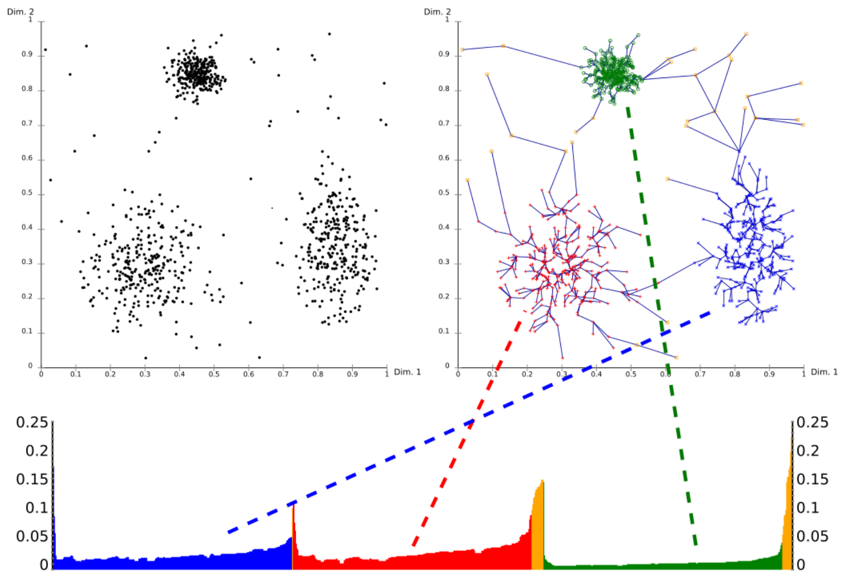


Figure C.3: For the synthetic data that can be seen in the top-left plot that has a label (blue,red and green), we observe how the reachability-plot (bottom plot) can be visualized from the top-right plot (constructed spanning tree). Since this data is easily clusterable, one can see in the reachability-plot that there are distinct outliers (marked in yellow) that separates the valleys in between. One can think that the deeper a valley is, the tighter the cluster. For this example everything that gets a higher reachability score than 0.1 is assigned as noise.

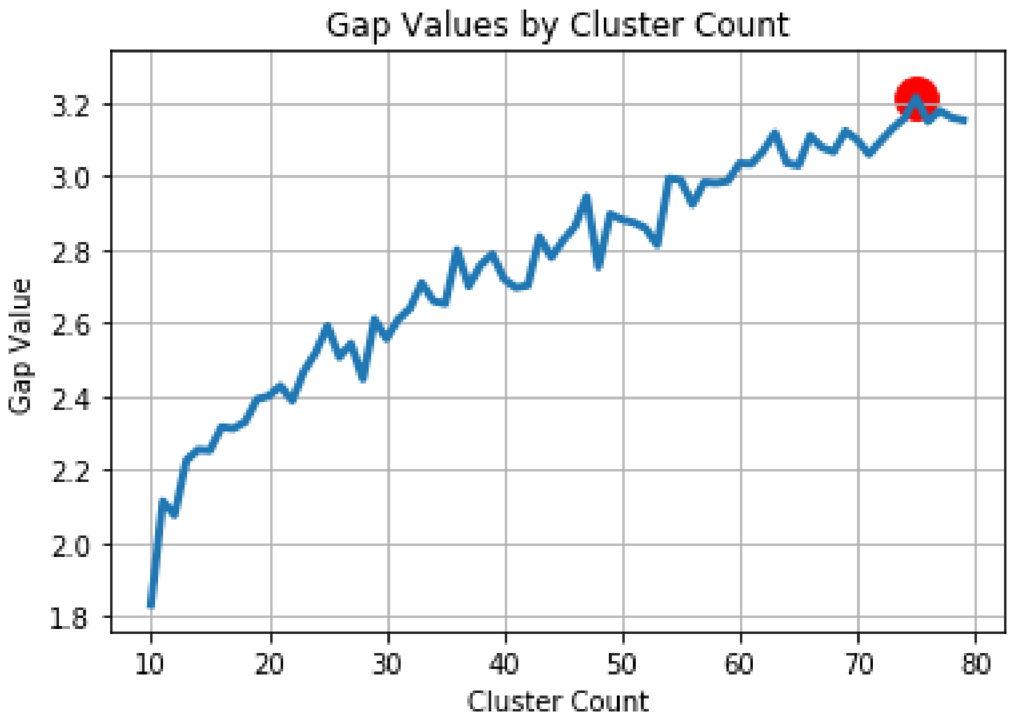


Figure C.4: Graph showing the gap statistic over number of clusters. The red dot represents the ideal amount of clusters based on the gap value.

D
Appendix 4: Complete result tables

This appendix includes tables with more detailed evaluations of the results from section 3. The tables show the evaluation metrics every model.

Table D.1 refers to section 3.1 where dummy and Random Forest classifier were evaluated.

Table D.2 refers to section 3.2 where different gradient boosting models were trained and evaluated.

Table D.3 refers to section 3.4 where XGBoost classifiers were trained on a subset of the data set, adding data from an influx database and configuration features as environmental features in two steps.

configuration	**dummy** F1 weighted	F1 macro	F1 anomalies	**random forest** F1 weighted	F1 macro	F1 anomalies
packet_link_capacity	0.1294	0.0236	0.4891	0.7521	0.6213	0.1833
input_alarm_threshold	0.4814	0.0641	0.6477	0.904	0.3998	0.0185
atpc_min_output_power	0.4501	0.03	0.0274	0.7866	0.4463	0.1132
atpc_max_output_power	0.279	0.0248	0.354	0.7885	0.4421	0.0411
max_output_power	0.0504	0.014	0.0577	0.9114	0.8354	0.0392
min_output_power	0.8937	0.1926	0.2727	1.0	1.0	0.7159
e1_number	0.9631	0.1975	0.0	0.9756	0.447	0.0
tx_freq	0.0226	0.0112	0.2105	0.9988	0.9978	0.7113
channel_spacing	0.5257	0.1591	0.8986	0.9895	0.863	0.3155
traffic_type	0.9619	0.329	0.0	0.9756	0.6404	0.0
ber_alarm_threshold	0.6487	0.43	0.0	0.9112	0.8788	0.1509
tx_oper_status	0.9015	0.4828	0.0606	0.9676	0.8557	0.0606
modulation	0.0947	0.0244	0.6538	0.6876	0.5371	0.1099
max_modulation	0.1332	0.0378	0.463	0.9055	0.7486	0.1519
average	**0.4668**	**0.1443**	**0.2954**	**0.8967**	**0.6938**	**0.1865**

Table D.1: Evaluation of our baseline models: Dummy Classifier and Random Forest Classifier, see section 3.1.

configuration	**XGBoost**			**LightGBM**			**CatBoost**		
	F1 weighted	F1 macro	F1 anomalies	F1 weighted	F1 macro	F1 anomalies	F1 weighted	F1 macro	F1 anomalies
packet_link_capacity	0.8	0.7444	0.2538	0.776	0.685	0.262	0.743	0.6253	0.1709
input_alarm_threshold	0.9054	0.4005	0.0312	0.9128	0.4355	0.0507	0.9044	0.3934	0.0
atpc_min_output_power	0.8414	0.6097	0.0676	0.8403	0.5906	0.0806	0.7913	0.3966	0.0702
atpc_max_output_power	0.8252	0.5738	0.0364	0.8127	0.5447	0.0189	0.7771	0.4033	0.0564
max_output_power	0.9232	0.8572	0.0338	0.9182	0.8565	0.0324	0.9013	0.8262	0.033
min_output_power	1.0	1.0	0.7159	1.0	1.0	0.5671	1.0	1.0	0.796
e1_number	0.9771	0.4986	0.0	0.9754	0.4529	0.0	0.9756	0.4659	0.0
tx_freq	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
channel_spacing	0.9893	0.8788	0.5965	0,9898	0,8947	0,2948	0.988	0.8465	0.3155
traffic_type	0.9771	0.6839	0.0	0,9761	0,6325	0.0	0.9765	0.6477	0.0
ber_alarm_threshold	0.9211	0.8931	0.1509	0,9139	0,8832	0,25	0.9088	0.8762	0.1852
tx_oper_status	0.9672	0.8541	0.0625	0,9696	0,8697	0,0606	0.9709	0.8757	0.0606
modulation	0.7314	0.5669	0.6087	0,7307	0,5622	0,193	0.7189	0.5108	0.1435
max_modulation	0.9356	0.9047	0.7643	0,9409	0,9182	0,7079	0.9255	0.8013	0.2559
average	**0.9139**	**0.7476**	**0.3087**	**0,9112**	**0,7376**	**0,2513**	**0.8987**	**0.6906**	**0.2205**

Table D.2: Evaluation of different gradien boosting algorithms: XGBoost, LightGBM and Catboost. See section 3.2.

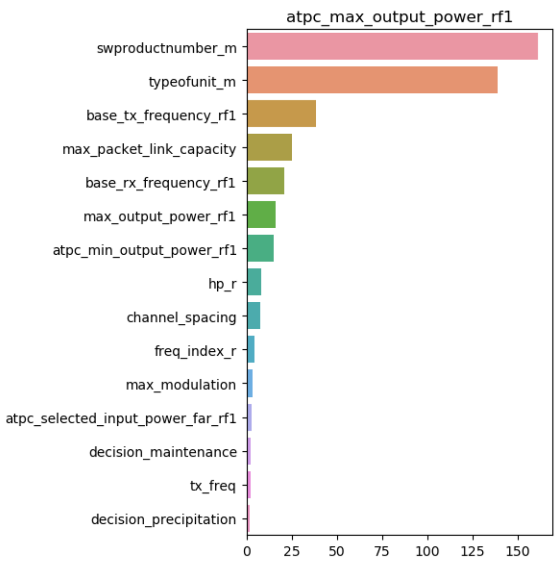
configuration	**no influx**			**influx**			**influx and configurations**		
	F1 weighted	F1 macro	F1 anomalies	F1 weighted	F1 macro	F1 anomalies	F1 weighted	F1 macro	F1 anomalies
packet_link_capacity	0.8237	0.6679	0.3177	0.8979	0.82	0.4812	0.924	0.8454	0.4763
input_alarm_threshold	0.9425	0.3791	0.0	0.9485	0.3348	0.0	0.9724	0.4044	0.0
atpc_min_output_power	0.9282	0.5965	0.1481	0.9649	0.8592	0.1429	0.994	0.9862	0.8926
atpc_max_output_power	0.8948	0.7102	0.4218	0.9407	0.8657	0.6031	0.9767	0.9755	0.889
max_output_power	0.9583	0.8717	0.208	0.9643	0.9219	0.0364	0.9791	0.9723	0.0364
min_output_power	1.0	1.0	1.0	1.0	1.0	99.0	0.9993	0.9965	99.0
e1_number	0.9756	0.2479	0.0	0.9744	0.2477	0.0	0.995	0.5642	0.0
tx_freq	1.0	1.0	0.9559	0.9993	0.9994	0.8739	0.9993	0.998	0.8739
channel_spacing	0.9993	0.9985	1.0	1.0	1.0	0.8907	1.0	1.0	0.8907
traffic_type	0.973	0.4953	99.0	0.975	0.5618	99.0	1.0	1.0	99.0
ber_alarm_threshold	0.9183	0.8941	0.0	0.9282	0.9071	0.2222	0.962	0.9504	0.2222
tx_oper_status	0.9647	0.5888	0.0	0.9647	0.5888	0.0	0.9926	0.9323	0.6154
modulation	0.6919	0.4391	0.4445	0.7637	0.5447	0.2007	0.87	0.636	0.7065
max_modulation	0.9403	0.8565	0.636	0.9504	0.9101	0.7504	0.9785	0.965	0.8577
average	**0.9293**	**0.6961**	**0.3948**	**0.948**	**0.7544**	**0.3501**	**0.9745**	**0.8733**	**0.5384**

Table D.3: Evaluating XGBoost classifier on a subset of data with and without influx data and configuration features as environment. See section 3.4.

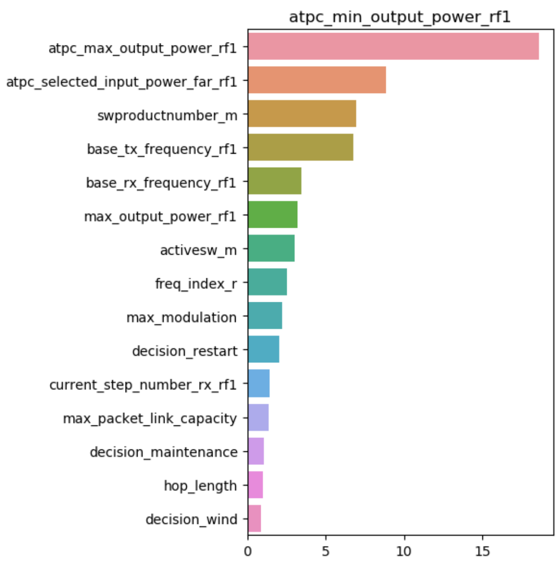
E

Appendix 5: Feature importance plots

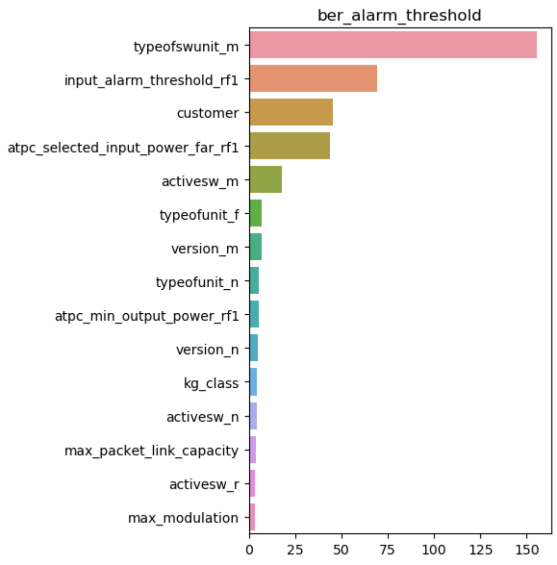
The following plots show the gain feature importance for every XGBoost model resulting from the run described in section 3.4. The gain feature importance describes the impact of a feature in the decision making of a model compared to all other features. The values should therefore be interpreted relatively to the other values.



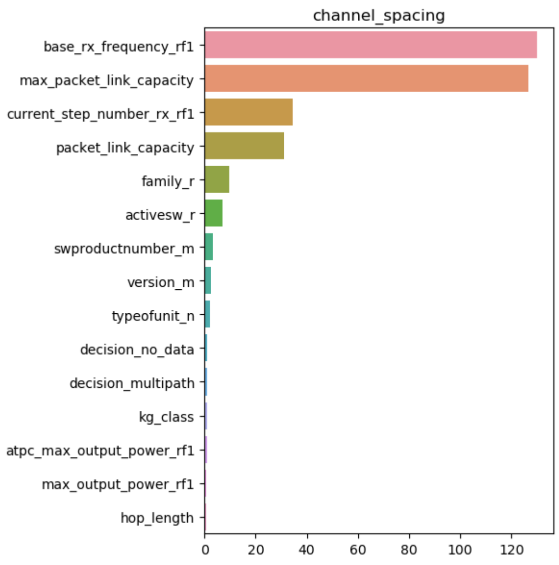
(a) maximum output power of automatic transmit power control



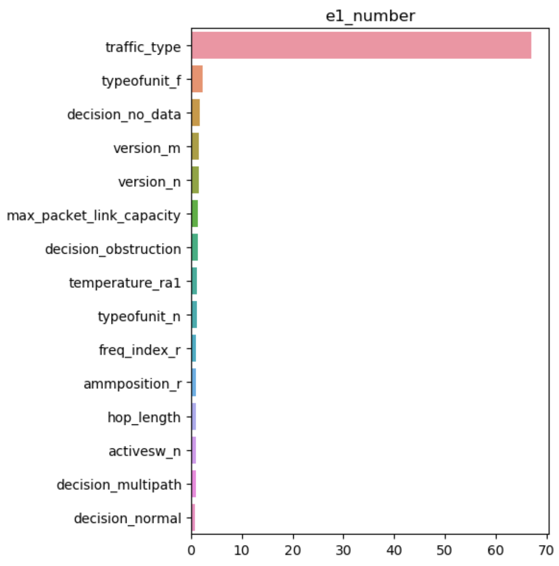
(b) minimum output power of automatic transmit power control



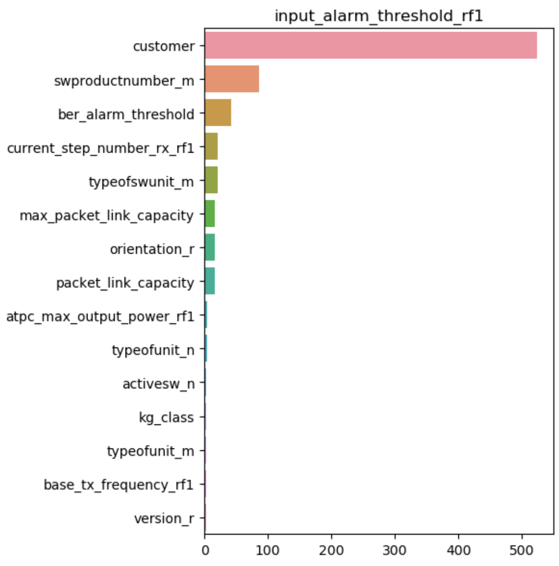
(c) alarm threshold for the bit-error-rate



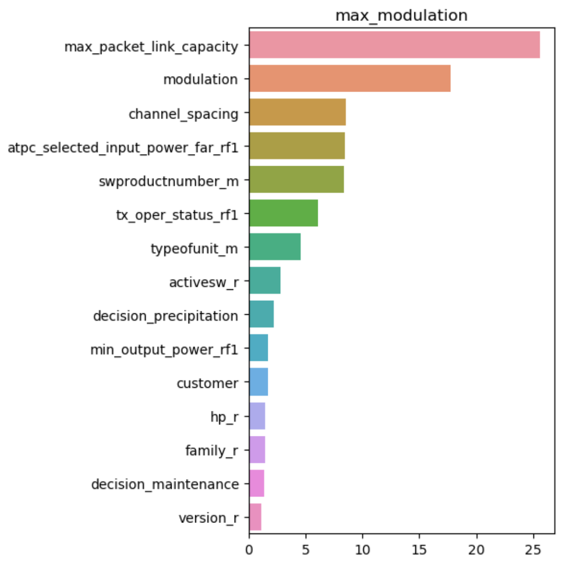
(d) channel spacing



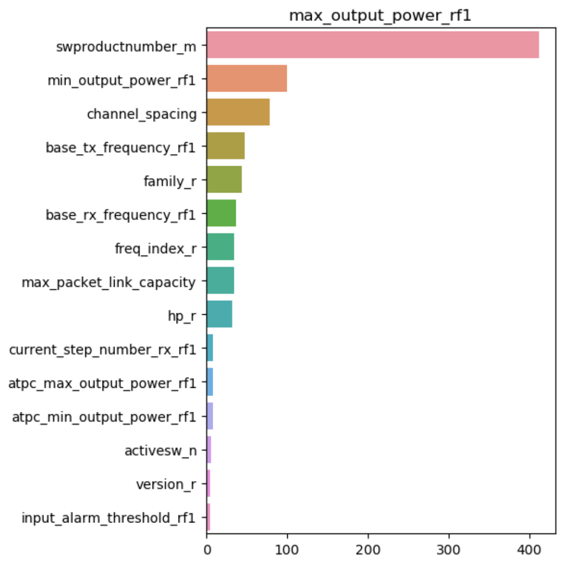
(e) e1 number



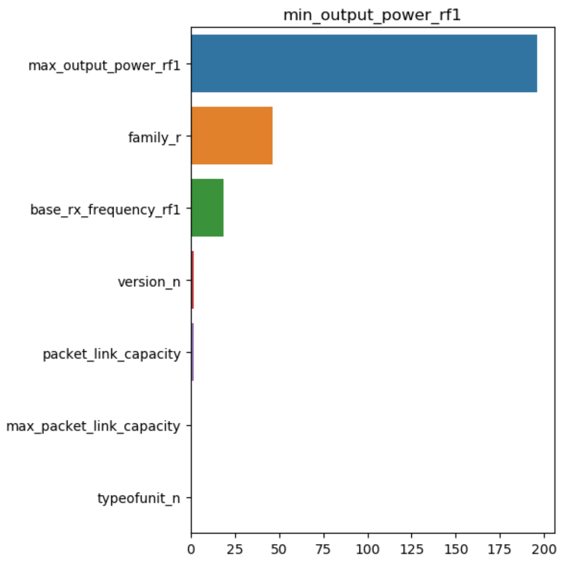
(f) alarm threshold for input signal



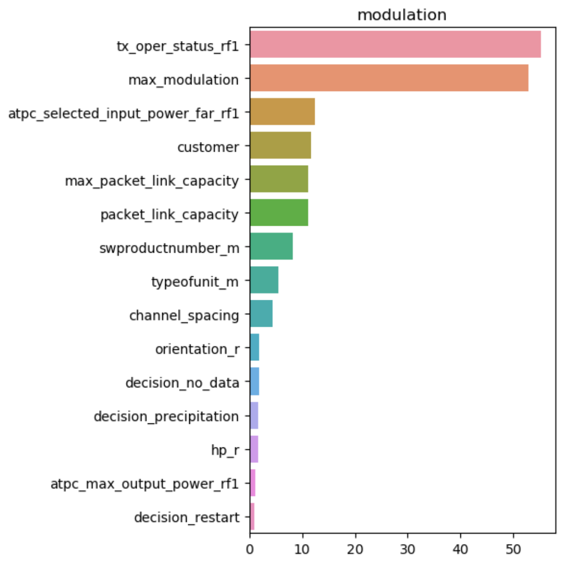
(g) maximum modulation



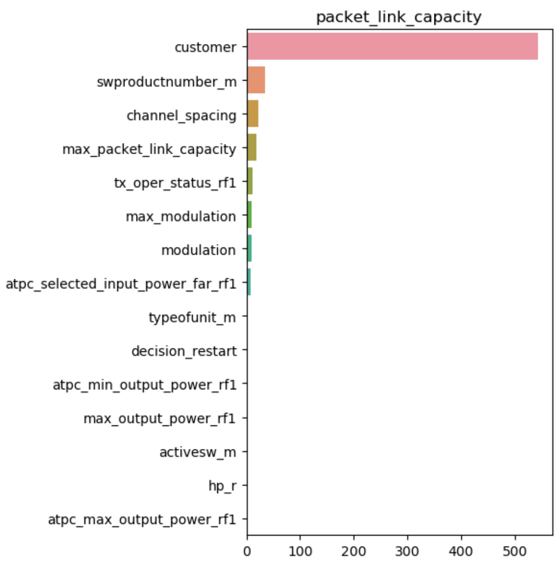
(h) maximum for output power



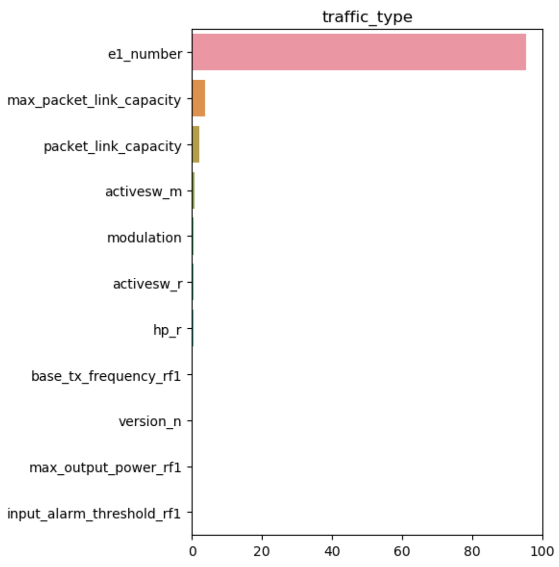
(i) minimum for output power



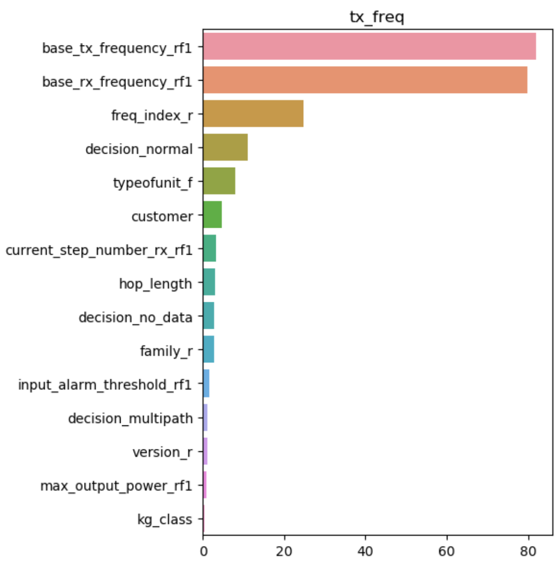
(j) modulation



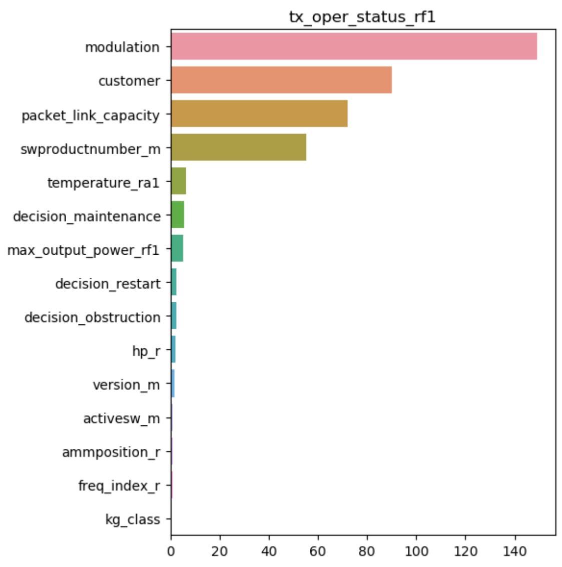
(k) packet link capacity



(l) traffic type



(m) tx frequency



(n) tx operation status

Figure E.-2: Feature importance plots for all models (one model for each configuration feature