On Designing a Machine Learning Based Wireless Link Quality Classifier

Gregor Cerar*†, Halil Yetgin*‡, Mihael Mohorčič*†, Carolina Fortuna*

*Department of Communication Systems, Jožef Stefan Institute, SI-1000 Ljubljana, Slovenia.
†Jožef Stefan International Postgraduate School, Jamova 39, SI-1000 Ljubljana, Slovenia.
‡Department of Electrical and Electronics Engineering, Bitlis Eren University, 13000 Bitlis, Turkey.
{gregor.cerar | halil.yetgin | miha.mohorcic | carolina.fortuna}@ijs.si

Abstract—Ensuring a reliable communication in wireless networks strictly depends on the effective estimation of the link quality, which is particularly challenging when propagation environment for radio signals significantly varies. In such environments, intelligent algorithms that can provide robust, resilient and adaptive links are being investigated to complement traditional algorithms in maintaining a reliable communication. In this respect, the data-driven link quality estimation (LQE) using machine learning (ML) algorithms is one of the most promising approaches. In this paper, we provide a quantitative evaluation of design decisions taken at each step involved in developing a ML-based wireless LQE on a selected, publicly available dataset. Our study shows that, re-sampling to achieve training class balance and feature engineering have a larger impact on the final performance of the LQE than the selection of the ML method on the selected data.

Index Terms—link quality estimation, machine learning, data-driven optimization, data preprocessing, feature selection.

I. INTRODUCTION

Machine learning (ML) is becoming an increasingly popular way of solving various aspects in communications in general and wireless networks in particular. Data driven link quality estimation (LQE) techniques where the researchers manually developed models have been proposed over the last two decades [1]–[3]. More recently, the manual model development is being automated, by using ML algorithms that approximate the distribution of the underlying random variable and are thus able to learn the quality of a link [4], [5].

LQE developed using ML can estimate the quality of a link in a continuous value space, in this case the ML performs a regression [4], [6]–[10]. Alternatively, if they estimate the quality in a discrete value space, the ML performs classification [5], [11]–[13]. By analyzing the existing body of work developing classification models for LQE, we notice the following approaches: binary, two class or multi-class.

The first type is a *binary or a two-class output*, which is produced by the classification model. This type of output can be found in [3], [11], [12], [14], [15]. The applications noticed are mainly (binary) decision making [3] and above/below threshold estimation [11], [12], [14], [15].

The second type is *multi-class output* value. Similar to the first type, it is also produced by the classification model. The multi-class output values are utilized in [5], [13], [16]–[19], where [17], [19] use a three-class, [16] utilizes a four-class, [13], [18] rely on a five-class, and [5] leverages a seven-class output. The applications observed are the categorization and estimation of the future LQE state, which is expressed through labels/classes. It is not always clear from the related work how the authors select the number of classes. However, according to [20], a wireless link seems to follow a non-linear S-shaped curve with three regions. All works using a three class output model seem to consider this characteristic of the link.

For developing ML models in any application area, generally some very precise steps that are well established in the community are followed [21], [22], namely data preprocessing, model building and model evaluation. The data preprocessing stage is known to be the most time-consuming process and tends to have a major influence on the final performance of the model. This stage includes several steps such as data cleaning and interpolation, feature selection and re-sampling. While most of the identified research developing LQEs explicitly mention aspects of cleaning and interpolation and feature selection, none evaluate the impact of the design decision taken at these steps on the final performance of the ML-based LQE. However, the majority evaluate the impact of the ML method selection on the final performance of the ML-based LQE.

Additionally, none of the works mentions aspects of the re-sampling step, that is particularly critical for the generalization capability of a model. Re-sampling is used in ML communities when the available input data is imbalanced [23], [24]. For instance, assume a classification problem where the aim is to classify links into *good*, *bad* and *intermediate* classes, similar to the problem approached in [17], [19]. If the *good* class would represent 75% of the examples in the training dataset, *bad* would represent 20% and *intermediate* would represent the remaining 5%, then a ML model would likely be well trained to recognize the *good* as it has been exposed to many such instances, however it might have difficulties in recognizing the other two minority classes.

In this paper, we aim to show the impact of design decisions taken at each step of the process of designing a ML-based LQE model on the final performance of the model. To realize our aim, we first select the Rutgers publicly available dataset [25]

and a decision tree[1] as a representative ML-based classification model. Then, we systematically perform each step of the knowledge discovery process [21] on the selected dataset using the selected model, meanwhile varying the design parameters at each step. This way, we are able to systematically quantify the influence of each of the design steps on the final performance of our classifier, therefore providing an in-depth step by step understanding on the process of learning to classify wireless links.

The main contributions of this paper are:

- A systematic quantification of the influence of the design steps on the final performance of our wireless link quality classifier is provided. The highlights of the quantification are that, for the chosen problem and dataset, the generation of synthetic features from the only available training feature received signal strength indicator ($RSSI$), yields up to 6% higher accuracy and is able to better discriminate the intermediate class up to 49%. The choice of ML method has less, relatively smaller impact on the final model performance with all the selected algorithms yielding an accuracy performance between 94% and 95% and minority class is detected between 87% and 89%.

- A first time evaluation of the impact of re-sampling on wireless link quality classification is realized using ML. In the case of the chosen imbalanced dataset, by using standard re-sampling, the minority class is correctly detected in over 87% of the instances, yielding more than 25 percentage points increase in the performance and comes at a small 2% decrease in overall accuracy.

The remainder of the paper is organized as follows. Section II elaborates on the selected dataset and Section III analyses the importance of the cleaning and interpolation steps applied to the selected dataset. Then, the importance of feature engineering, window selection and re-sampling strategy is emphasized in Section IV, while Section V examines the influence of the model selection on the final performance of the LQE classifier. Then, potential generalization perspectives of the proposed models and findings for other datasets are discussed in Section VI. Finally, Section VII concludes the paper.

II. RUTGERS DATASET SUMMARY

The Rutgers trace-set [25] includes 4,060 distinct link traces, which are gleaned from 812 unique links with 5 different noise levels, i.e., 0, -5, -10, -15 and -20 dBm. Readily available trace-set features include raw RSSI, sequence numbers, source node ID, destination node ID and artificial noise levels. In this particular experiment, we observe that the packets are sent every 100 milliseconds for a period of 30 seconds. Therefore, each trace is composed of 300 packets. Besides, based on the specifications of the radio used, each RSSI value ranges from 0 to 128, where the value 0 represents a bad link with no signal and 127 indicates a good link with strong signal, while observed value of 128 represents an

error. Nonetheless, a statistical analysis of the Rutgers trace-set reveals that 960 link traces out of 4,060 (23.65%) are entirely empty indicating no packets were received, and that a total of 1,218,000 packets were sent and only 773,568 (63.51%) were correctly received.

All the scripts developed for the comparative performance analyses are publicly available [2] for researchers to reproduce, re-use on other data-sets and improve upon our analyses.

III. ANALYSIS OF CLEANING & INTERPOLATION STEPS

The first step of the process of developing a ML-based LQE involves data cleaning and interpolation. The reason for that is because models that are automatically created using ML algorithms can be significantly biased as a result of invalid and missing data. First of all, a valid time series corresponding to each link has to be extracted, which is referred to as a series of ordered tuples each of which contains a packet sequence number and corresponding measured link metrics. The obtained values in the tuples have to be within valid ranges. For instance, the sequence numbers have to be identical with the packets sent during the trace collection, and the values of the link metrics have to remain within the valid ranges that are specified by the transceiver data sheets. Roughly speaking, link metrics with regard to the received radio signals, i.e., RSSI and link quality indicator (LQI) can be extracted directly from the hardware registers of the corresponding transceivers, whereas link metrics concerning packet data transmission, i.e., packet reception ratio (PRR) and packet success rate (PSR) are computed with suitable software procedures.

As described in Section II, the Rutgers trace-set contains invalid values and a considerable number of missing sequence numbers due to the lost packets. Most of the available out-of-the-box data mining algorithms cannot handle these invalid values, e.g., *NaN* and $\pm\infty$ of IEEE 754 standard, or they are simply ignored. To quantify the impact of selected cleaning and interpolation approaches to the final performance of the model, we assume the use of a decision tree algorithm trained with a trio of instant RSSI, averaged RSSI and standard deviation RSSI values, stratified k-fold[3] and pruning[4], standard normalization, and random oversampling (ROS) approach, as discussed in Sections IV-A, IV-B and IV-C.

From the perspective of data preprocessing steps for ML models, there are many approaches for handling missing data [26], [27]. To reveal the impact of the approach to missing values on link quality classification, we train the same model, i.e., decision trees, stratified k-fold and pruning, along with the same feature set for the following cases; a) without handling the missing values, b) using a simple time series approach where we interpolate missing data with Gaussian noise, and c) with the aid of domain knowledge. In the case

---

[1]Decision tree is one of the predictive modeling approaches utilized in ML.

[2]ML LQE scripts: https://github.com/sensorlab/link-quality-estimation

[3]K-fold cross-validation is a statistical procedure of splitting data into training and test chunks used for evaluating the ability of ML models.

[4]Pruning is a technique used for shrinking the size of decision trees by discarding sections of the tree that may be too specific and thus lead to over-fitting.

of interpolation with Gaussian noise, gaps of missing data are filled with random values based on the previous and next valid values. Regarding domain knowledge, we replace the missing RSSI values with 0, which represents a poor quality link with no received signal, yielding PRR equal to 0. Recalling that possible RSSI values are integers ranging between 0 (bad link with no signal) and 127 (good link with strong signal), while observed value of 128 represents an error.

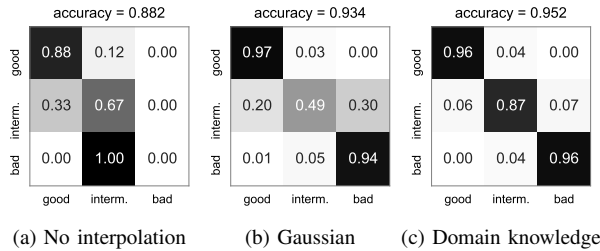| (a) No interpolation | (b) Gaussian | (c) Domain knowledge |

Fig. 1. Different interpolation cases with a nonlinear decision tree algorithm and ROS.

Fig. 1 presents the relative performance of the models for all three interpolation cases using the form of a confusion matrix[5], i.e., indicating how well the model classifies individual instances. The better the classifier the darker the diagonal of the confusion matrix and the whiter the non-diagonal squares. For this particular case, the best performing model in terms of accuracy (95.2%) is the model using domain knowledge in Fig. 1c. *While the difference in accuracy between the best two models is approximately 2 percentage points, their respective confusion matrices indicate that the model using interpolation with domain knowledge is superior as it better discriminates between the three link types.* This comparison confirms that [11], [13], [30] took the best design decision by using domain knowledge for cleaning and interpolation.

IV. ANALYSIS OF FEATURE ENGINEERING

The second step of the process of developing a ML-based LQE involves feature engineering. The feature engineering step may involve several sub-steps depending on application requirements, type of data and type of ML problem. For our purpose of learning to classify LQE, we distinguish three sub-steps discussed in the following subsection.

A. Analysis of feature selection

Feature selection is the process of selecting relevant raw features and/or creating synthetic features to be used for training ML models. When the number of possible input features is very large, then usually only the most relevant ones are selected to be used for the model. On the other hand, when the number of possible input features is very low, then creating synthetic features starting from the available ones to aid in better model development is employed. Feature selection is a

fundamental step and can be performed manually or, in some cases, can be built automatically by existing algorithms, such as support vector machines (SVMs)[6].

To analyze and understand the influence of feature selection on the model performance, we consider a set of standard feature engineering procedures on the selected dataset. The Rutgers trace-set has only two available attributes useful for LQE, i.e., the instant (raw) RSSI value and the sequence number, therefore exploring synthetic feature generation for model improvement seems to be the only feasible option for this step. The sequence number is leveraged for the computation of PRR which represents the target value, therefore leaving only RSSI as possible training feature. This classification obeys the following rules:

$$y = f(\text{PRR}) = \begin{cases} \text{bad}, & \text{if PRR} \leq 0.1 \\ \text{intermediate}, & \text{otherwise} \\ \text{good}, & \text{if PRR} \geq 0.9, \end{cases} \quad (1)$$

$$\mathbf{y} = [y_1, y_2, \ldots, y_n], \quad \forall y \in \{\text{bad, intermediate, good}\}. \quad (2)$$

A typical approach in ML for such limited trace-sets is to investigate whether synthetic features, such as average RSSI over a time window or polynomial interactions [31][7], can aid in training to acquire more accurate models compared to that of the instant RSSI values. Fig. 2 shows the influence of the best-performing feature combinations on the classification performance. For this analysis, we assume interpolation based on domain knowledge, i.e., replacing missing values with zeros, as discussed in Section III. Additionally, synthetic feature creation with prediction window size W_{PRR} and historical window size $W_{history}$ are set to 10, while utilizing standard normalization and ROS approach, as discussed in Sections IV-C and IV-B. In this analysis we predict the link quality as per Eq. (1) for the next prediction window W_{PRR}. Noting that the windows W_{PRR} and $W_{history}$ are utilized for computing link quality labels and features, respectively.

We can see from the results listed in Fig. 2(a) that the decision tree based model, trained using stratified k-fold and pruning, that uses the only available feature, $RSSI$, yields 89% accuracy and 38% correctly identified *intermediate* class, we can call this the baseline performance. *The best performing feature combination that uses two synthetically generated features in $RSSI_{avg}$, $RSSI_{std}$ addition to RSSI yields an accuracy of 95.2% and 87% correctly identified intermediate class as can be seen in Fig. 2(l).* Moreover, Fig. 2(j) also shows that $RSSI_{avg}$ alone yields great results, i.e., 93% accuracy and 87% correctly identified *intermediate* class. Positive powers of $RSSI$ have no major advantage over the baseline as can be seen from Fig. 2(c), (d) and (e), while negative powers

---

[5]Confusion matrix is a table layout, with rows for the instances of a predicted class and columns for the instances of an actual class, used for the problem of statistical classification in order to exhibit the performance of an algorithm. Readers are referred to [28] and [29] for further details.

[6]Support Vector Machine is a prominent ML algorithm seeking solutions for both classification and regression problems.

[7]Polynomial interactions are employed for generating new features from the already available ones with the intention of improving the performance of ML model.
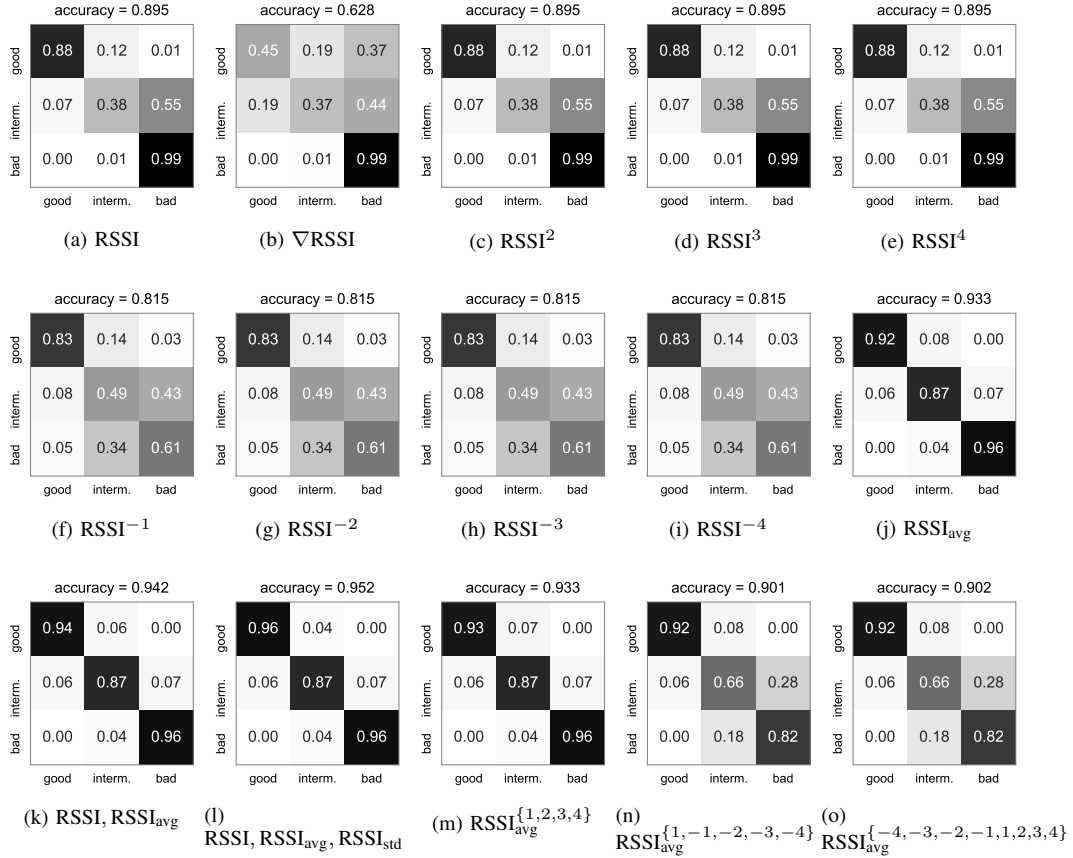
Fig. 2. The influence of feature selection on the performance of the nonlinear model using decision trees.

lower overall accuracy, albeit they perform better then the baseline for the *intermediate* class and significantly worse for the *bad* class as per Fig. 2(f), (g), (h) and (i). In the last row of the table, Figs. 2(k-o) show that other synthetic combinations of $RSSI_{avg}$ perform relatively better than the baseline. *As a conclusion, it can be seen that the generation of synthetic features from the only available training feature RSSI, yields up to 6% higher accuracy and is able to discriminate the intermediate class up to 49% better.*

B. Analysis of window selection

For examining the influence of the window selection on the performance of the model, we need to distinguish between two types of windows. The first one is the historical window $W_{history}$ that is used for computing features such as $RSSI_{avg}$. The second one is the prediction window W_{PRR} that is used for computing the link quality labels. The majority of related works mention details about the window selection step. However, many of them fail to specify the size of the window used for the models they propose and evaluate. Additionally, the window size tends to be smaller for more reactive or online models, such as in [6], [12], while for less reactive models, as proposed in [5], [18], the window size is likely larger.

Given that the investigated Rutgers trace-set consists of 300 packets per link, the size limits for the two windows are
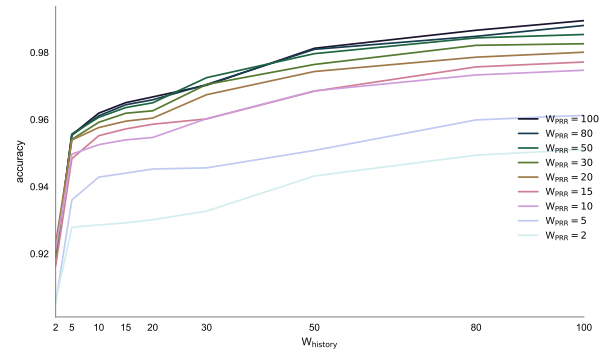

Fig. 3. Overview of the influence of a discrete set of window sizes on the accuracy of the proposed nonlinear model.

within $[0, 300]$ packets, where opting for the value 0 indicates no windowing and favoring the value 300 suggests per link labeling. Therefore, we restrict the range of the window sizes to $[2, 100]$ packets, within which we investigate the performance with a discrete set of nine values $\{2, 5, 10, 15, 20, 30, 50, 80, 100\}$. In this analysis, we predict the link quality for the next prediction window $PRR(W_{PRR})$ considering the Rutgers trace-set with domain knowledge interpolation, the decision tree algorithm, with stratified k-fold and pruning,

the feature vector (RSSI, $\text{RSSI}_{\text{avg}}(W_{\text{history}})$, $\text{RSSI}_{\text{std}}(W_{\text{history}})$), standard normalization and the ROS approach.

As portrayed in Fig. 3, the best performing model is the one utilizing $W_{\text{PRR}} = 100$, which predominantly outperforms the models based on other W_{PRR} settings, although all results for window size above 30 are rather similar.

The results, in general, reveal that; (i) a longer historical window improves prediction because there is more information about how the link performed in the past, and (ii) increasing the prediction window (computing the future value of the classes for link quality) also leads to an improvement of the accuracy. Both observations, however, can also be a side-effect of "smoothing"/averaging data from a relatively static trace-set. More explicitly, larger prediction windows are unable to inform on short-term effects, although they can help better in identifying the overall link behavior. It is worth noting that the optimal combination of values for historical and prediction windows is data dependent, however, the trade-offs discussed in this section can be adopted for general models. While the Rutgers trace-set is relatively static, for a more dynamic trace-set the optimal window sizes are likely smaller.

To develop a suitable LQE model, the agility of the model has to be specified by the designer considering dynamically changing environments, e.g., for designing a routing algorithm in a largely mobile wireless network. Additionally, the practical memory limitations of the devices have to be taken into account when developing a suitable LQE model. This is mainly because more agile estimators use smaller window sizes, and therefore they tend to consume less memory, and yet yield low accuracy. Even though larger window sizes assist in attaining high accuracy, the cold start period, during which the historical window is initialized, leads to an estimation delay.

C. Re-sampling strategy

From the analysis of the actual values in the considered Rutgers trace-set, it can be readily observed that there are 61% *good*, 34% *bad* and only 5% *intermediate* class entries. This distribution of data is largely imbalanced due to the artifact of the experiment, where the nodes were close to each other and the interference level was relatively low. Therefore, the majority of the links were actually good as expected and this was not due to the missing values within one particular class category of link quality. Additionally, it has been acknowledged in the literature [20] that the *intermediate* region of the receivers tends to be relatively narrow compared to the *good* and *bad* regions, and therefore this naturally forms a scarcely populated class for intermediate regions in such trace-sets, yet having an important influence to ML-based LQE models although, as mentioned in the introduction as part of the motivation for this work, this aspect has been neglected by all the related work we have reviewed.

Imbalanced trace-sets are often encountered in ML and data mining communities and they are typically dealt with an appropriate re-sampling strategy. For studying the influence of the re-sampling strategy on the performance of the model for link quality classification, we employ the standard ROS and
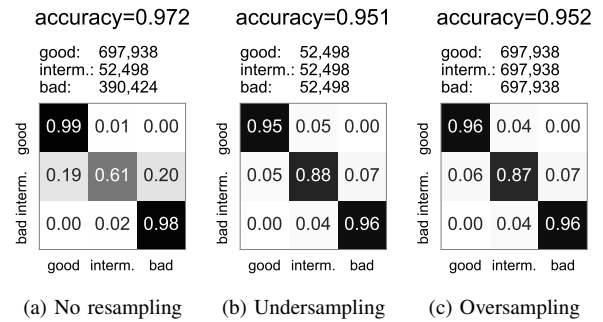

Fig. 4. Different re-sampling strategies on the pipeline with a standard normalization and nonlinear decision tree algorithm using (RSSI, RSSI_{avg} and RSSI_{std} features.

the random undersampling (RUS) approaches. The ROS [23], [24] approach equalizes all class sizes to the size of the majority class by duplicating the trace-set entries of the minority classes; therefore the resulting re-sampled dataset becomes larger. On the contrary, the RUS [23], [24] approach equalizes all class sizes to the size of the minority class by randomly discarding instances from other larger classes. Hence, the new resampled dataset becomes smaller. It is observed from the numbers of good, intermediate and bad links in Figs. 4(b) and (c) that with both approaches, i.e., ROS and RUS, we are able to acquire a training dataset with balanced classes, about 50k examples for each class of RUS and 690k examples for ROS.

Fig. 4 illustrates that re-sampling strategies on the Rutgers trace-set decrease the overall accuracy of the classification model from 97.2% to slightly above 95%. Some more advanced re-sampling strategies [32] may limit this decrease in performance. However, when no re-sampling is performed, the minority class, i.e., *intermediate* is only correctly detected in 61% of the instances, indicating that the model is over-fitted to the majority of the classes. *In the case of re-sampling, the minority class is correctly detected in over 87% of the instances, yielding more than 25 percentage points increase in the performance.* This improvement comes at a relatively small performance cost for the majority classes, inducing 3-4 percentage points decline for the *good* links and 2 percentage points reduction for the *bad* links.

Considering this analysis, we may hint that, in the case of [13], where the performance of the predictor on two of the five classes is modest, employing a resample strategy might lead to better discrimination of those classes. Re-sampling may also improve other proposed estimators, for example the ones in [5], [11], [18], [30].

The results for the selected Rutgers trace-set reveal that there is no significant distinction between the two re-sampling strategies, i.e., RUS and ROS. This is likely due to the relatively large size of the intermediate class. Although the intermediate class only represents 5% of the population, it still contains more than 52,000 samples. However, looking beyond this particular trace-set, the RUS approach may suffer from excluding a certain number of majority class instances

and may affect the representativeness of the remaining data points, especially for more dynamic trace-sets. On the other hand, due to the enlarged number of data points, the ROS approach requires more computing resources for building a model. Note that the results obtained in this section are based on interpolation and cleaning using domain knowledge, instant RSSI, $RSSI_{avg}$ and $RSSI_{std}$ as features and W_{PRR} and $W_{history}$ of size 10.

V. ANALYSIS OF MODEL SELECTION

The final step of this systematic analysis is concerned with the influence of the ML algorithm selection on the performance of LQE models. To provide a comparative analysis of the impact, we examine logistic regression and linear SVM as representatives of linear ML algorithms, and decision trees, random forests and a multilayer perceptron[8], as representatives of nonlinear model. As a baseline reference model, we leverage the majority classifier, which in our case, classifies all links in the *good* class.

(a) Majority classifier   (b) Logistic regression   (c) Decision Trees

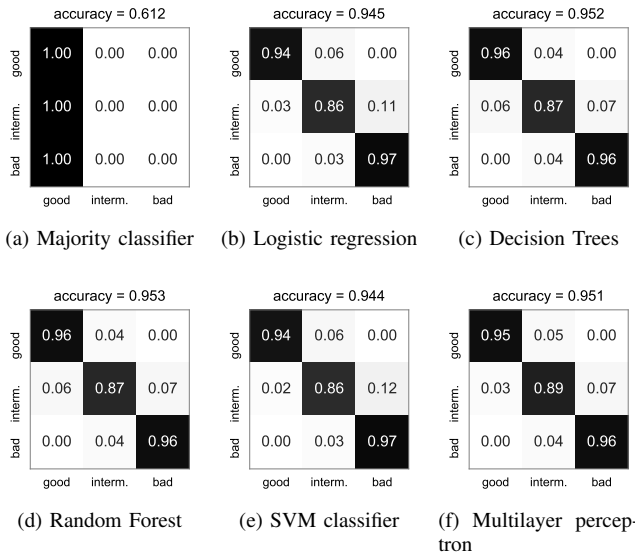(d) Random Forest   (e) SVM classifier   (f) Multilayer perceptron

Fig. 5. The influence of the choice of ML algorithm on the effectiveness of LQE models.

The analysis in this section is conducted by using domain knowledge interpolation, the feature vector consisting of instant RSSI, $RSSI_{avg}$ and $RSSI_{std}$, windowing with $W_{PRR} = 10$, $W_{history} = 10$, and a ROS approach over the Rutgers trace-set. The selected ML algorithms are evaluated using 10-times stratified K-fold cross-validation [33], [34]. Note that we obey the rule of cumulative parameterization throughout the data preprocessing steps in order to reveal the impact of each step on the ML algorithms for the sake of the LQE model proposed.

Fig. 5 shows that all the selected ML models apart from the reference majority classifier have comparable performance, with an accuracy above 94%. Decision tress, random forests

---

[8]Multilayer perceptron is a class of feed-forward neural networks, where information travels through a directed non-cyclic graph of computational units called perceptrons (artificial neurons).

and multi-layer perceptrons, non-linear ML models, are very similar at 95% accuracy. SVM with linear kernel and logistic regression are then at 94%. Slightly lower performance of the linear models such as logistic regression and SVM conforms to the findings in the literature that LQE is a nonlinear function [7], [10], [18], [20]. Looking at the ability of the ML algorithms to identify the minority class, the multilayer perceptron outperforms all the other ML algorithms considered for this analysis.

One of our major observation from the analysis of ML-based LQE models is that the slightly better performance of nonlinear ML-based LQE models to the linear counterparts conforms to the findings in the state-of-the-art literature as it can be observed in [7], [10]. Besides, upon the conclusions drawn in [6], [11], [12], [30], which are mainly compared to 4B [35], we can see that ML-based LQE models consistently outperform the traditional analytical estimators.

VI. THREATS TO VALIDITY

In this paper, we quantify the influence of various design steps of the ML process on the final results for the case of LQE estimation on a single dataset. However, by using alternative datasets, the general conclusions that the preprocessing steps have a relatively higher influence on the final result than the model selection would still hold for any other dataset, albeit the exact numbers will differ from the ones presented herein. Consider linear regression as a representative of linear ML models and decision trees as a representative of non-linear ML models. Roughly speaking, it is expected that both models will perform well when learning to approximate linear problems, but that the decision trees will perform significantly better with non-linear problems. However, with the aid of data preprocessing, more specifically feature interactions, non-linearities can be captured by the engineered features. When fed into the linear model, these engineered features will compensate for the shortcomings of the model and the final performance will be comparable to the non-linear model that typically benefits less from feature interactions.

Similarly, both in statistics and ML, various re-sampling approaches are leveraged for a better understanding of the underlying distribution that generated the available observations. Without sufficient training examples, ML models are unable to learn about certain classes, therefore using re-sampling for balancing out the datasets to eventually attain superior results on minority classes is essential. This is also a generally applicable conclusion, albeit the expected improvement in performance will be strictly dependent on the dataset.

VII. CONCLUSIONS

In this paper, we provided a systematic quantification of the influence of the design steps on the final performance of a wireless link quality classifier. Among others, we found that, for the chosen problem and dataset, the generation of synthetic features from the only available training feature $RSSI$, yields up to 6% higher accuracy and is able to discriminate the intermediate class up to 49% better. The choice of ML method

has relatively smaller impact on final model performance with all the selected algorithm yielding accuracy between 94% and 95% and minority class is detected between 87% and 89%.

We also provided a first time evaluation of the impact of re-sampling on wireless link quality classification using ML. In the case of the chosen imbalanced dataset, by using standard re-sampling, the minority class was correctly detected in over 87% of the instances, yielding more than 25 percentage points increase in the performance and comes at a small decrease in accuracy that can be mitigated with more advanced re-sampling techniques.

ACKNOWLEDGMENT

REFERENCES

[1] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in Winter Simulation Conf., California, USA, 8-11 December 1996, pp. 597–604.

[2] H. Balakrishnan and R. H. Katz, "Explicit loss notification and wireless web performance," in IEEE Globecom Internet Mini-Conference, Sydney, Australia, November 1998, http://nms.lcs.mit.edu/~hari/papers/globecom98/.

[3] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in Proceedings of the 1st international conference on Embedded networked sensor systems, California, USA, 5–7 November 2003, pp. 14–27.

[4] W. Sun, W. Lu, Q. Li, L. Chen, D. Mu, and X. Yuan, "WNN-LQE: Wavelet-Neural-Network-Based Link Quality Estimation for Smart Grid WSNs," IEEE Access, vol. 5, pp. 12 788–12 797, July 2017.

[5] S. Demetri, M. Zúñiga, G. P. Picco, F. Kuipers, L. Bruzzone, and T. Telkamp, "Automated estimation of link quality for LoRa: A remote sensing approach," in ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'19), Montreal, CA, 15-18 April 2019.

[6] T. Liu and A. E. Cerpa, "Foresee (4C): Wireless link prediction using link features," in 10th Int. Conference on Information Processing in Sensor Networks (IPSN'10), Chicago, USA, 12-14 April 2011.

[7] P. Millan, C. Molina, E. Medina, D. Vega, R. Meseguer, B. Braem, and C. Blondia, "Time series analysis to predict link quality of wireless community networks," Computer Networks, vol. 93, no. 2, pp. 342–358, Dec. 2015.

[8] E. Ancillotti, C. Vallati, R. Bruno, and E. Mingozzi, "A reinforcement learning-based link quality estimation strategy for RPL and its impact on topology management," Computer Communications, vol. 112, pp. 1–13, Nov. 2017.

[9] H. Okamoto, T. Nishio, M. Morikura, K. Yamamoto, D. Murayama, and K. Nakahira, "Machine-learning-based throughput estimation using images for mmwave communications," in 2017 IEEE 85th Vehicular Technology Conference (VTC Spring). IEEE, 2017, pp. 1–6.

[10] M. L. Bote-Lorenzo, E. Gómez-Sánchez, C. Mediavilla-Pastor, and J. I. Asensio-Pérez, "Online machine learning algorithms to predict link quality in community wireless mesh networks," Computer Networks, vol. 132, pp. 68–80, February 2018.

[11] T. Liu and A. E. Cerpa, "Temporal adaptive link quality prediction with online learning," ACM Transactions on Sensor Networks (TOSN), vol. 10, no. 3, p. 46, 2014.

[12] S. Rekik, N. Baccour, M. Jmaiel, and K. Drira, "Low-power link quality estimation in smart grid environments," in International Wireless Communications and Mobile Computing Conference (IWCMC'15), Dubrovnik, Croatia, 24-28 August 2015.

[13] X. Luo, L. Liu, J. Shu, and M. Al-Kali, "Link quality estimation method for wireless sensor networks based on stacked autoencoder," IEEE Access, vol. 7, pp. 21 572–21 583, 2019.

[14] N. Baccour, A. Koubâa, M. B. Jamâa, D. Do Rosario, H. Youssef, M. Alves, and L. B. Becker, "Radiale: A framework for designing and assessing link quality estimators in wireless sensor networks," Ad Hoc Networks, vol. 9, no. 7, pp. 1165–1185, September 2011.

[15] Z.-Q. Guo, Q. Wang, M.-H. Li, and J. He, "Fuzzy logic based multidimensional link quality estimation for multi-hop wireless sensor networks," IEEE Sensors Journal, vol. 13, no. 10, pp. 3605–3615, October 2013.

[16] C. A. Boano, M. Zuniga, T. Voigt, A. Willig, and K. Römer, "The triangle metric: Fast link quality estimation for mobile wireless sensor networks," in International Conference on Computer Communication Networks, Zurich, Switzerland, 2-5 August 2010.

[17] W. Rehan, S. Fischer, and M. Rehan, "Machine-learning based channel quality and stability estimation for stream-based multichannel wireless sensor networks," MDPI Sensors, vol. 16, no. 9, p. 1476, Sept. 2016.

[18] J. Shu, S. Liu, L. Liu, L. Zhan, and G. Hu, "Research on link quality estimation mechanism for wireless sensor networks based on support vector machine," Chinese Journal of Electronics, vol. 26, no. 2, pp. 377–384, April 2017.

[19] H.-J. Audéoud and M. Heusse, "Quick and efficient link quality estimation in wireless sensors networks," in Wireless On-demand Network Systems and Services (WONS), 2018 14th Annual Conference on. IEEE, 2018, pp. 87–90.

[20] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: A survey," ACM Transactions on Sensor Networks (TOSN), vol. 8, no. 4, p. 34, September 2012.

[21] U. Fayyad, G. P.-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," AI magazine, vol. 17, no. 3, p. 37, 1996.

[22] M. Kulin, C. Fortuna, E. De Poorter, D. Deschrijver, and I. Moerman, "Data-driven design of intelligent wireless networks: An overview and tutorial," Sensors, vol. 16, no. 6, p. 790, 2016.

[23] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," ACM SigKDD Explorations Newsletter, vol. 6, no. 1, pp. 1–6, June 2004.

[24] B. W. Yap, K. A. Rani, H. A. A. Rahman, S. Fong, Z. Khairudin, and N. N. Abdullah, "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets," in Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng'13) - Lecture Notes in Electrical Engineering, December 2014, pp. 13–22.

[25] S. K. Kaul, M. Gruteser, and I. Seskar, "Creating wireless multi-hop topologies on space-constrained indoor testbeds through noise injection," in TRIDENTCOM, Barcelona, Spain, 1-3 March 2006.

[26] J. M. Jerez, I. Molina, P. J. García-Laencina, E. Alba, N. Ribelles, M. Martín, and L. Franco, "Missing data imputation using statistical and machine learning methods in a real breast cancer problem," Artificial intelligence in medicine, vol. 50, no. 2, pp. 105–115, 2010.

[27] R. J. Little and D. B. Rubin, Statistical analysis with missing data. John Wiley & Sons, 2014, vol. 333.

[28] W. Castro, J. Oblitas, M. De-La-Torre, C. Cotrina, K. Bazn, and H. Avila-George, "Classification of cape gooseberry fruit according to its level of ripeness using machine learning techniques and different color spaces," IEEE Access, vol. 7, pp. 27 389–27 400, 2019.

[29] R. A. Alshinina and K. M. Elleithy, "A highly accurate deep learning based approach for developing wireless sensor network middleware," IEEE Access, vol. 6, pp. 29 885–29 898, 2018.

[30] T. Liu and A. E. Cerpa, "TALENT: temporal adaptive link estimator with no training," in Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, Toronto, Canada, November 2012, pp. 253–266.

[31] A. A. Freitas, "Understanding the crucial role of attribute interaction in data mining," AI Review, vol. 16, no. 3, pp. 177–199, Nov. 2001.

[32] L. Lusa et al., "Smote for high-dimensional class-imbalanced data," BMC bioinformatics, vol. 14, no. 1, p. 106, 2013.

[33] S. Arlot, A. Celisse et al., "A survey of cross-validation procedures for model selection," Statistics surveys, vol. 4, pp. 40–79, March 2010.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., "Scikit-learn: Machine learning in Python," Journal of machine learning research, vol. 12, no. Oct, pp. 2825–2830, 2011.

[35] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, "Four-bit wireless link estimation," in ACM SIGCOMM Sixth Workshop on Hot Topics in Networks (HotNets-VI'07), Atlanta, Georgia, 14-15 November 2007.