2024 13th International Conference on Control, Automation and Information Sciences (ICCAIS) | 979-8-3315-4204-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCAIS63750.2024.10814234

Machine Learning in Wireless Link Estimation: A Comparative Study of Supervised, Unsupervised, and Deep Learning Approaches

Hoang-Long Nguyen[1,2,*],Ngoc-Nhat Huynh[1,2,*], Huy-Tan Thai[1,2], Khanh-Hoi Le-Minh[1,2]

[1]University of Information Technology, Ho Chi Minh City, Vietnam
[2]Vietnam National University, Ho Chi Minh City, Vietnam
Email: {21522304, 21522416}@gm.uit.edu.vn, {tanth, hoilmk}@uit.edu.vn
*: The authors have the same contribution

***Abstract*—The proliferation of Internet of Things devices demands reliable wireless connections against negative environmental factors and signal interference. In this paper, we present a comprehensive comparative analysis of about 17 machine learning algorithms with different settings, including unsupervised, supervised, and deep learning models, for wireless link estimation. We evaluate the accuracy and adaptability of these algorithms under various connection conditions in two datasets: the publicly available dataset from Colorado and a custom dataset collected using Raspberry Pi 4 devices. The key finding reveals that while deep learning models achieve superior performance, they are more prone to overfitting than traditional machine learning approaches. Notably, unsupervised models struggle to find meaningful cluster structures in complex datasets but achieve high accuracy on simpler ones. Based on this finding, researchers in this field can have insights into the strengths and limitations of machine learning approaches, offering a practical foundation for developing more effective and adaptable algorithms for wireless link estimation in diverse IoT environments.**

***Index Terms*—Wireless link estimation, Machine learning, Deep learning, Network performance evaluation.**

I. Introduction

A. Context of wireless link estimation

The growth of the Internet of Things (IoT) and wireless sensor devices deployed extensively across urban and rural environments, has been notable in recent years. However, in wireless networks, signal radio is affected by various temporal and spatial factors, presenting significant challenges for consistent communication. Thus, robust wireless link estimation techniques are essential for dynamically adjusting radio link parameters in response to changing environmental conditions [1]. In detail, wireless connection estimation is the process of determining and evaluating the quality of the communication channel between a transmitter and receiver within a wireless system. This process typically involves measuring and analyzing signals to estimate channel characteristics such as signal strength, noise level, and delay. It then adjusts transmission parameters to enhance signal quality, minimize errors, and optimize resource utilization. However, wireless link estimation faces significant challenges mainly due to channel variation. This is because the transverse channels are often affected by interference. Signal damping and environmental vibrations make the estimation more complicated.

B. Motivation for using machine learning in wireless estimation

To address the challenges in wireless link estimation, various methodologies have been proposed to enhance accuracy and adaptability. Traditional estimation methods [2], [3] tend to rely on mathematical models created to describe the communication channels, which can be challenging when dealing with complex and rapidly changing conditions in challenging environments. This often results in low urgency and poor productivity in real-world scenarios. To address these limitations, machine learning (ML) has emerged as a promising solution. Machine learning algorithms could effectively extract complex patterns from data, enabling more flexible and adaptive estimation models. Unlike traditional methods, they could learn not only complicated network features but also their relationship to identify the impact of environmental conditions on the signal. Therefore, machine learning algorithms offer more flexible and adaptive approach to identifying and responding to high-impact factors, which are based on supervised learning [4]–[8] and unsupervised learning [9], [10]. Regarding supervised ML, the author in [4] proposed a model based on Gradient Boosting Decision Trees (GBDT) to assess the link quality, classified into four labels: good, average, bad, and very bad. Instead of using models to evaluate link quality, the authors in [5] used models to predict link quality. This approach enabled the detection of link breaks or routing changes before packet loss occurred. Their model was evaluated based on two network features: RSSI and Packet Delivery Ratio (PDR). For unsupervised ML, Gregor et al. [9] introduced a process for developing models to detect link exceptions. By sorting and selecting the results with the best accuracy, they achieved up to 96% accuracy depending on the exception type. The authors

979-8-3315-4204-7/24/$31.00 ©2024 IEEE

in [10] constructed an efficient algorithm, namely kernel power density (KPD) to group multipath components (MPCs) by analyzing the density variations of the K most recent components.

C. Overview of Existing Works and Research Gaps

Although ML has been applied to link estimation in many studies, a comprehensive evaluation of various algorithms on multiple datasets remains lacking, particularly in unsupervised learning and changing channel conditions. Existing studies typically focus on applying specific machine learning algorithms to solve the link estimation problem, rather than providing a broader comparative analysis. Therefore, this study aims to bridge this gap by providing a comprehensive review of various machine learning algorithms, particularly focusing on unsupervised and supervised learning. This study not only expands the existing knowledge, but also provides deeper insights into potential solutions to address the current challenges in this research field.

D. Objectives and Contributions of the Paper

Light by the identified research gaps, this paper presents a systematic review of both supervised and unsupervised machine learning algorithms for wireless link estimation. The primary goal is to assess the classification performance of these algorithms across various datasets, providing a comprehensive comparative analysis. In sum, the contributions of this paper are as follows:

- We conduct a comparative analysis of machine learning algorithms for wireless link estimation, including various unsupervised, supervised, and deep learning models. This analysis aims to identify the most effective approaches for dealing with the complexities of real-world wireless environments.
- We comprehensively analyze the current limitations in machine learning-based wireless link estimation. It serves as a baseline for proposing future research directions and potential improvements in the application of machine learning algorithms to wireless link estimation.

II. Methodology

This paper employs traditional and deep learning models to conduct experiments on two datasets, described in Section II-B. The traditional machine learning algorithms, encompass both supervised and unsupervised approaches, as detailed in Section II-A1 and II-A2, respectively. The deep learning models are outlined in Section II-A3.

A. Algorithms

1) Supervised Machine learning Algorithms: As shown in Table I, several supervised algorithms and their hyperparameters are evaluated in our work for the wireless link estimation task. The details for each algorithm is described below:

TABLE I: Supervised ML algorithms are used to evaluate the datasets.

Algorithm	Hyper-parameter
Decision Trees	criterion = ['gini', 'entropy', 'log_loss']
Random Forest	criterion = ['gini', 'entropy', 'log_loss']
SVC	kernel= ['linear', 'poly', 'rbf', 'sigmoid']
KNN	algorithm = ['ball_tree', 'kd_tree']
Gradient Boosting	criterion = ['friedman_mse', 'squared_error']

a) Decision trees: are commonly used to solve classification and regression problems. This algorithm recursively divides the dataset into smaller subgroups based on features to create branching conditions, yielding individual predictions at each leaf [11]. This algorithm is highly interpretable due to its transparent rules.

b) Random forests: leverages the power of multiple decision trees to enhance accuracy and mitigate the risk of overfitting [12]. Unlike a single decision tree, Random Forests constructs multiple trees from random samples of data, each making its own prediction. Its final output is the average (in regression problems) or majority vote (in classification problems). This ensemble approach makes Random Forests more stable and less sensitive to noise in the data than individual decision trees.

c) SVC: focuses on finding an optimal hyperplane to maximize the distance between classes based on the nearest data points, known as "support vectors" [13]. SVCs are capable of handling complex classification problems through the use of kernels, which transform data into a higher-dimensional feature space. The main advantage of SVCs is their ability to perform efficiently on datasets with a small number of samples but a large number of features.

d) KNN: is a simple yet powerful machine learning algorithm that predicts a new data point's value based on its "K" nearest neighbors [14]. Despite its simplicity, KNN's performance can be affected by noisy data and is inefficient with computationally expensive when dealing with large datasets.

e) Gradient Boosting: is an ensemble technique that constructs a strong model by combining multiple weak learners, particularly decision tree [15]. It enhances performance by adding these learners sequentially to minimize a defined loss function through gradient descent optimization.

2) Unsupervised Machine Learning Algorithms: Table II lists unsupervised algorithms and its hyperparameter commonly used in machine learning tasks as described below:

a) Gaussian Mixture: is a probabilistic model assuming data points are generated from a combination of Gaussian distributions [16], while each distribution is assigned to a cluster. Though computationally fast and capable of assigning

TABLE II: Unsupervised ML algorithms are used to evaluate datasets.

Algorithms	Hyper-parameter
Gaussian Mixture	covariance_type = ['full, 'tied', 'diag', 'spherical']
Bayesian Gaussian Mixture	covariance_type = ['full, 'tied', 'diag', 'spherical']
K-Means	algorithm=['lloyd', 'elkan']
Bisecting K-Means	algorithm=['lloyd', 'elkan']
Agglomerative Clustering	linkage=['ward', 'complete', 'average', 'single']

data points to multiple clusters, GM can be resource-intensive for large datasets and may require multiple initializations to avoid local optima.

b) The Bayesian Gaussian Mixture (BGM): is a variant of GM that uses the Bayesian approach, which produces score estimates for all variables as well as uncertainty in the posterior distribution of these estimates [17]. In detail, Bayesian assumes a prior distribution for $\boldsymbol{\theta}$, a distribution that represents the degree of likelihood of each possible value of $\boldsymbol{\theta}$ before the data is observed. Then to deduce $\boldsymbol{\theta}$, BGM needs to consider the conditional distribution of $\boldsymbol{\theta}$. This represents the updated beliefs about $\boldsymbol{\theta}$ after incorporating the data.

c) K-Means: a classic hard clustering algorithm, definitively assigns data points to a single cluster [18]. This algorithm attempts to identify cluster centers and assigns data points based on their proximity, often using Euclidean distance as a measure of similarity. K-Means does not guarantee that the output of cluster centers is the same for each run because it depends on the cluster centers being initialized.

d) Bisecting K-Means: is a hybrid approach based on K-Means and hierarchical clustering, aiming for improved cluster quality in less time, particularly with large datasets [19]. It iteratively divides a cluster using K-Means with $k = 2$ until a predefined number of clusters is obtained. Compared to standard K-Means, it's computationally efficient for numerous clusters and less sensitive to initial conditions. However, both algorithms can suffer from converging to local minima.

e) Agglomerative Clustering: is one of the two commonly used clustering techniques. At first, this algorithm considers each data point as an individual cluster and gradually merges the two closet clusters into a new cluster until the desired number of clusters is satisfied. The distance between clusters can be calculated by various linkage criteria, such as ward or single.

3) Deep learning Models:

a) TabNet: is a Transformer-based model, designed for tabular data by Google Cloud AI [20]. It identifies the most relevant features at each step, incorporating the attention mechanism and decision tree algorithm to effectively process both continuous and categorical data. In addition, TabNet is self-explanatory, helping users better understand the model's decisions, and thereby increasing its reliability in practical applications.

b) Deep Neural Networks (DNNs): are a popular architecture in deep learning, consisting of multiple layers of interconnected neurons. When used for tabular data, DNNs face challenges due to the heterogeneity of features and potential nonlinear relationships between variables. Therefore, data preprocessing, including normalization and encoding of categorical variables, is important to achieve good performance.

c) Entity Embeddings: is a method that employs embedding techniques to represent categorical variables as numerical vectors, enabling neural networks to learn relationships between categorical values [21]. It helps reduce data dimensionality and enhances the model's performance when processing tabular data. By capturing hidden features within categorical variables, these embedding vectors contribute to improved model prediction accuracy.

d) TabTransformer: is based on the Transformer architecture and optimized for handling categorical variables in tabular data [22]. With the attention layer, TabTransformer finds complex relationships between categorical values and combines them with continuous features through densely connected layers. It is designed to scale and efficiently manage complex data. However, this also requires a high computational resource and needs to be carefully tuned to achieve the best performance.

e) Neural Oblivious Decision Ensembles (NODE): is a hybrid model between a neural network and a decision tree [23]. NODE uses layers of random decision trees during training to learn nonlinear relationships between variables, and combines them with neural networks to enhance the representation. This model has demonstrated high performance in many problems involving tabular data, even outperforming models such as XGBoost or Random Forest. However, training NODE can be complex and requires tuning many hyperparameters to achieve the best results.

f) DeepGBM: is a hybrid model, constructed from deep neural networks (DNNs) and Gradient Boosting Machine (GBM) models such as XGBoost [24]. This model utilizes GBMs to generate new features from the original data, which are then used as input for the neural network. This helps to take advantage of the generalization ability of DNNs and the nonlinear processing power of GBMs.

g) AutoInt (Automatic Feature Interaction Learning via Self-Attentive Neural Networks): employs the self-attention mechanism to automatically learn how features interact with each other in tabular data [25]. This model requires manual compilation of features or interactions between variables. AutoInt, capable of automatically learning complex relationships from data, proves particularly valuable in scenarios where inter-variable relationships are challenging to identify.

B. Dataset

This paper utilizes two datasets to benchmark the above machine learning and deep learning models.

The first dataset from [26], named data_col, comprises 579,203 samples and 14 features, such as x_coordinate, y_coordinate, tx_pow, rssi, rssi_mean. Irrelevant features, including file_name, _coordinate, y_coordinate, device_name and received, are removed from the dataset due to their lack of relevance to the learning process. Subsequently, missing categorical values are imputed using the "most-frequently" strategy, replacing them with the most commonly occurring categorical value. Next, one-hot encoding is applied to encode the categorical features before applying MixMax scaler to scale the entire dataset. The final data_col dataset contains 25,722 samples and 18 features, including labels.

The second dataset, data_rasp, employs a Raspberry Pi 4 device for data collection. The collection process comprises three main phases. At first, the Raspberry Pi 4 gathers Wi-Fi information. Then, it stores and extracts this information, saving it to a CSV file as feature values. The final stage merges all the sub-datasets into the final data_rasp dataset.

C. Evaluation metrics

To assess the classification performance of supervised MLs and deep learning models, accuracy, precision, recall, and F1 metrics are employed. To ensure robust evaluation, 10-fold cross-validation is implemented, allowing for the calculation of mean and standard deviation values for each metric. Regarding unsupervised MLs, the ARI (Adjusted Rand Index) metric is used to evaluate the similarity between the predicted cluster and the actual cluster. The ARI measures the similarity between predicted and actual clusters, ranging from -1 to 1.

III. Experiment

A. The classification performance of supervised machine learning algorithms

Table III presents the classification performance (accuracy, recall, precision, F1-score) of five supervised MLs on two datasets.

In the first dataset, namely data_col, the evaluation results reveal diverse performance across supervised ML algorithms. The Decision Tree (DT) and Gradient Boosting (GB) achieve perfect scores (100%) on all metrics, regardless of parameter configurations. This raises concerns about potential overfitting. The Random Forest (RF) algorithm demonstrates more reasonable performance, with accuracy and F1-score approximately 91% and 92%, respectively. The promising approach is the Support Vector Classifier (SVC), demonstrating the most stable performance with consistently high mean scores over 99% for linear, polynomial, and RBF kernels. In summary, while several models show high performance, the SVC stands out as the most reliable, balancing high accuracy with consistent results across folds.

In the second data set named data_rasp, the DT algorithm achieves high accuracy (above 92%) but exhibits lower precision and recall, resulting in a low F1-score at roughly 87%. The RF technique demonstrates similar results, achieving an accuracy of approximately 92% and an F1-score of 88%. In particular, SVC shows significant performance variability across different kernel types. The polynomial kernel yields the highest accuracy (98.38%) while the sigmoid kernel results in the lowest accuracy (89.27%). Notably, the KNN algorithm outperforms more complex models, reaching 98.57% accuracy with high stability in all metrics. In summary, the majority of algorithms achieve accuracy exceeding 85%, with KNN emerging as the top classifier. This consistently high performance across models suggests that the dataset may possess well-separated class structures with minimal overlap.

B. The clustering performance of unsupervised machine learning algorithms

Figure 1 shows the mean and standard deviation values of ARI of unsupervised algorithms on two experimental datasets.

The first subgraph, regarding the ARI for data_col dataset, reveals poor clustering performance across all algorithms, with mean ARI scores ranging from a mere 0.002 to 0.011. In detail, the Gaussian Mixture (GM) and Bayesian Gaussian Mixture (BGM) algorithms attain the highest mean ARI of 0.011. The K-Means and Bisecting K-Means exhibit an equal mean ARI of 0.008. Notably, the 'single' linkage parameter results in the lowest mean ARI of 0.002 among all algorithms. These consistently low ARI values indicate that none of the unsupervised algorithms could effectively identify the underlying cluster structure in this dataset.

The second subgraph, regarding the ARI for data_rasp dataset, demonstrates considerably improved clustering performance across all models, with mean ARI scores ranging from 0.768 to 0.882. Specifically, the GM algorithm achieves mean ARI values increasing from 0.783 to 0.789 across different settings. The BGM algorithm shows the most variation, ranging from 0.777 to 0.807 depending on the covariance type setting. Notably, the AC algorithm with the single linkage setting achieves the highest mean ARI reported of 0.881. The BKM algorithm shows consistent but lowest performance, with a mean ARI of 0.768. In summary, these results highlight the effectiveness of unsupervised models, especially AC with the single linkage setting, in capturing the underlying structure in this dataset.

C. The classification performance of deep learning models

Table IV shows the classification performance of the deep learning models on two experimental datasets.

On the data_col dataset, TabNet, DNN, DeepGBM, TabTrans, and NODE demonstrate high performance with accuracy, recall, and F1 scores of 99.9%. Besides, these models exhibit stability, as evidenced by their low standard deviation values. Meanwhile, on data_rasp, DNN, DeepGBM, TabTrans, and NODE yield the highest value of accuracy, recall, and F1 scores at 100%, with standard deviations approaching zero. AutoInt model has experienced a similar trend as these previous models. This result suggests that these models may

TABLE III: Mean and standard deviation values of accuracy, recall, precision, and F1 metrics of supervised MLs on two experimental datasets.

Models	data_col				data_rasp			
	Accuracy	Recall	Precision	F1	Accuracy	Recall	Precision	F1
Decision Tree (criterion='gini')	100.0 (±0.0)	100.0 (±0.0)	100.0 (±0.0)	100.0 (±0.0)	93.13 (±5.23)	89.66 (±8.02)	90.11 (±9.84)	87.96 (±9.85)
Decision Tree (criterion='entropy' | 'log_loss')	100.0 (±0.0)	100.0 (±0.0)	100.0 (±0.0)	100.0 (±0.0)	92.5 (±6.42)	89.13 (±8.93)	90.7 (±9.26)	87.41 (±10.8)
Random Forest (criterion='gini')	91.98 (±13.46)	92.48 (±12.7)	95.8 (±6.04)	92.72 (±12.51)	91.71 (±5.53)	87.94 (±8.45)	86.61 (±11.07)	86.33 (±10.14)
Random Forest (criterion='entropy' | 'log_loss')	91.47 (±12.76)	92.04 (±11.75)	93.97 (±8.41)	91.82 (±12.41)	92.56 (±5.06)	89.27 (±7.65)	88.63 (±9.72)	88.24 (±9.00)
SVC (kernel='linear')	99.90 (±0.06)	99.91 (±0.05)	99.91 (±0.05)	99.91 (±0.05)	94.98 (±5.06)	92.41 (±7.74)	93.83 (±7.09)	91.22 (±9.49)
SVC (kernel='poly')	98.73 (±3.12)	98.75 (±3.08)	98.72 (±3.15)	98.70 (±3.21)	98.38 (±3.06)	97.53 (±4.68)	98.53 (±2.52)	97.36 (±5.27)
SVC (kernel='rbf')	99.90 (±0.06)	99.91 (±0.05)	99.91 (±0.05)	99.91 (±0.05)	94.70 (±4.63)	92.12 (±7.13)	92.91 (±7.17)	91.11 (±8.65)
SVC (kernel='sigmoid')	22.37 (±2.63)	22.28 (±2.98)	21.95 (±5.02)	19.61 (±3.48)	89.27 (±4.93)	85.00 (±7.53)	85.68 (±9.04)	84.21 (±7.61)
KNN ('ball_tree' | 'kd_tree' | 'brute')	82.37 (±8.27)	81.29 (±8.42)	81.58 (±8.34)	80.92 (±8.76)	98.57 (±1.77)	97.85 (±2.72)	98.39 (±1.78)	97.88 (±2.69)
Gradient Boosting (criterion='friedman_mse' | 'squared_error')	100.0 (±0.00)	100.0 (±0.00)	100.0 (±0.00)	100.0 (±0.00)	94.04 (±5.48)	90.93 (±8.31)	93.39 (±8.40)	89.32 (±10.26)

TABLE IV: Mean and standard deviation values of Accuracy, Recall, Precision, and F1 metrics of deep learning models on two experimental datasets.

Models	data_col				data_rasp			
	Accuracy	Recall	Precision	F1	Accuracy	Recall	Precision	F1
TabNet	99.9 (±0.1)	99.9 (±0.1)	99.9 (±0.1)	99.9 (±0.1)	97.4 (±4.0)	97.4 (±4.0)	98.0 (±3.1)	97.3 (±4.2)
DNN	99.9 (±0.1)	99.9 (±0.1)	99.9 (±0.1)	99.9 (±0.1)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)
EM	95.4 (±1.5)	95.4 (±1.5)	95.4 (±1.5)	95.3 (±1.5)	87.0 (±0.7)	86.9 (±0.6)	86.9 (±0.6)	86.9 (±0.6)
TabTrans	99.9 (±0.1)	99.9 (±0.1)	99.9 (±0.1)	99.9 (±0.1)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)
NODE	99.9 (±0.00)	99.9 (±0.00)	99.9 (±0.00)	99.9 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)
DeepGBM	99.9 (±0.00)	99.9 (±0.00)	99.9 (±0.00)	99.9 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)
AutoInt	99.2 (±1.8)	99.2 (±1.8)	99.3 (±1.5)	99.2 (±1.8)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)

be overfitting the data_rasp dataset. The EM model has the lowest performance on both datasets, with 95.4% on data_col and 87.0% on data_rasp. It is observed that the performance on the data_col dataset is better than the data_rasp dataset. It is because the data_rasp has a simpler structure that these models readily learn and give the highest score in the training phase, which might leads to an overfitting situation in real-world scenarios. Additionally, this dataset might lack samples to adequately train these deep learning models.

IV. Conclusion

In this study, we provide a comprehensive comparative analysis of machine-learning algorithms for wireless link estimation. This analysis shows that large and complex datasets such as data_col are suitable for sophisticated machine-learning algorithms and deep-learning techniques, achieving high accuracy. However, this dataset does not have a clear cluster structure, so the results of the cluster algorithms are extremely low, with an ARI mean of approximately 0. Considering the data_rasp, while this dataset is effective for classification models and clustering models, it encounters an overfitting situation with the deep learning models. These findings are crucial for designing wireless link estimation systems in IoT networks, suggesting data characteristics when selecting algorithms.

Acknowledgments

This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number C2023-

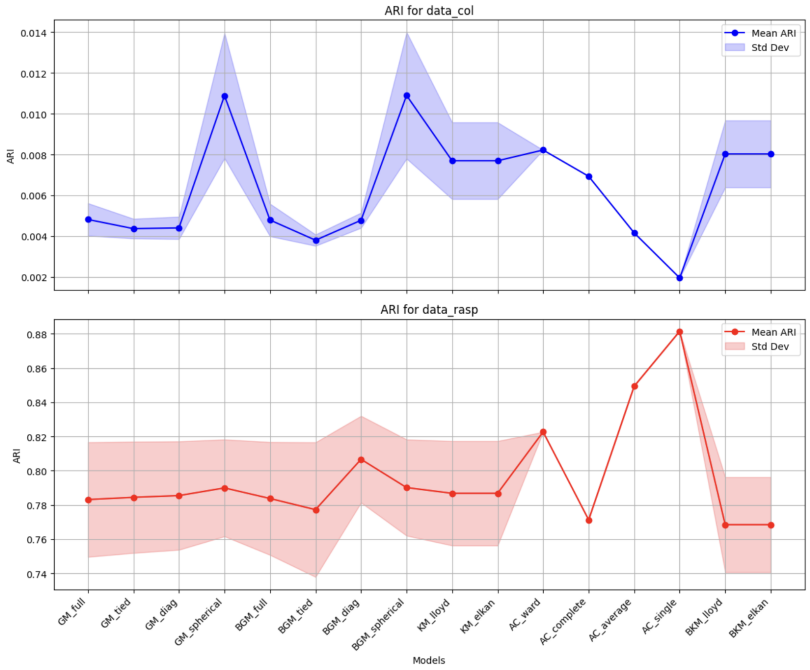

Fig. 1: The mean and standard deviation values of ARI metric of unsupervised models on two experimental datasets.

26-05.

REFERENCES

[1] 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, pp. 1–33, 2012.

[2] R. Fonseca, O. Gnawali, K. Jamieson, and P. A. Levis, "Four-bit wireless link estimation.," in *HotNets*, 2007.

[3] M. Senel, K. Chintalapudi, D. Lal, A. Keshavarzian, and E. J. Coyle, "A kalman filter based link quality estimation scheme for wireless sensor networks," in *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, pp. 875–880, IEEE, 2007.

[4] K.-H. Le-Minh, K.-H. Le, and Q. Le-Trung, "A lightweight machine-learning based wireless link estimation for iot devices," in *2022 27th Asia Pacific Conference on Communications (APCC)*, pp. 526–531, 2022.

[5] M. L. F. Sindjoung and P. Minet, "Estimating and predicting link quality in wireless IoT networks," *Annals of Telecommunications*, vol. 77, pp. 253–265, June 2022.

[6] Srinikethan Madapuzi Srinivasan, Tram Truong-Huu, and Mohan Gurusamy, "Machine Learning-Based link fault identification and localization in complex networks," *IEEE Internet of Things Journal*, 2019.

[7] Xionghui Luo, Xionghui Luo, Linlan Liu, Linlan Liu, Linlan Liu, Jian Shu, Jian Shu, Jian Shu, Manar Al-Kali, Jian Shu, and Manar Al-Kali, "Link quality estimation method for wireless sensor networks based on stacked autoencoder," *IEEE Access*, 2019.

[8] J. Piuma and P. Rattin, "Medición de caudales en canales de aforo mediante sistema telemétrico," in *2018 IEEE 9th Power, Instrumentation and Measurement Meeting (EPIM)*, pp. 1–7, 2018.

[9] Gregor Cerar, Gregor Cerar, Halil Yetgin, Halil Yetgin, Carolina Fortuna, and Carolina Fortuna, "Machine Learning-Based model selection for anomalous wireless link detection," *International Conference on Software, Telecommunications and Computer Networks*, 2021.

[10] R. He, Q. Li, B. Ai, Y. L.-A. Geng, A. F. Molisch, V. Kristem, Z. Zhong, and J. Yu, "A kernel-power-density-based algorithm for channel multipath components clustering," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7138–7151, 2017.

[11] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth, 1986.

[12] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[13] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[14] T. Cover and P. Hart, "Nearest-neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[15] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[16] Josep Lluís Solé and Josep Lluís Solé, "Book review: Pattern recognition and machine learning. cristopher m. bishop. information science and statistics. springer 2006, 738 pages," *Sort-statistics and Operations Research Transactions*, 2007.

[17] J. Lu, "A survey on bayesian inference for gaussian mixture model," 2021.

[18] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for k-means clustering," *Pattern Recognition Letters*, vol. 25, pp. 1293–1302, Aug. 2004.

[19] K.Abirami and Dr. P.Mayilvahanan, "Performance analysis of K-Means and bisecting K-Means algorithms in weblog data," 2016.

[20] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," *CoRR*, vol. abs/1908.07442, 2019.

[21] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," *CoRR*, vol. abs/1604.06737, 2016.

[22] X. Huang, A. Khetan, M. Cvitkovic, and Z. S. Karnin, "Tabtransformer: Tabular data modeling using contextual embeddings," *CoRR*, vol. abs/2012.06678, 2020.

[23] S. Popov, S. Morozov, and A. Babenko, "Neural oblivious decision ensembles for deep learning on tabular data," *CoRR*, vol. abs/1909.06312, 2019.

[24] G. Ke, Z. Xu, J. Zhang, J. Bian, and T.-Y. Liu, "Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 384–394, 2019.

[25] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "Autoint: Automatic feature interaction learning via self-attentive neural networks," *CoRR*, vol. abs/1810.11921, 2018.

[26] K. Bauer, D. Mccoy, B. Greenstein, D. Grunwald, and D. Sicker, "Physical layer attacks on unlinkability in wireless lans," vol. 5672, pp. 108–127, 08 2009.