5G Network Slice Type Classification using Traditional and Incremental Learning

Mohamad Ahmadinejad
Department of Computer Science
University of Regina
Regina, SK, Canada
mha808@uregina.ca

Tahmina Azmin
Department of Computer Science
University of Regina
Regina, SK, Canada
tah437@uregina.ca

Nashid Shahriar
Department of Computer Science
University of Regina
Regina, SK, Canada
nashid.shahriar@uregina.ca

NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium | 978-1-6654-7716-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/NOMS56928.2023.10154385

***Abstract*—The Fifth generation (5G) mobile network is expected to provide high bandwidth, low latency, and rapid user connectivity. 5G Mobile operators are seeking an effective solution that would enable them to support heterogeneous use cases with different Quality of Service (QoS) requirements by utilizing the existing physical infrastructure. 5G supports Network Slicing (NS), an end-to-end (E2E) logical network that is mutually isolated, has independent control, and can be managed independently. By slicing the network, mobile operators can effectively manage several network instances over a single infrastructure to provide a variety of applications, use cases, and business services while satisfying heterogeneous QoS requirements. With the advancement of Machine Learning (ML), future communication networks will need to use data-driven decision-making to achieve desired network performance. In this paper, we demonstrated a prediction mechanism using Machine and Deep Learning Algorithms in traditional and incremental ways to select the suitable network slice for various user requirements and device types. Using a publicly available dataset and Incremental Learning model called Stochastic Gradient Descent (SGD), we successfully classified incoming user requests to appropriate network slices with an accuracy of 99.33%.**

***Index Terms*—Network Slicing, 5G Cellular Networks, Machine Learning, Deep Learning, Incremental Learning**

I. Introduction

In the past two decades, the growth of mobile devices has been spurred by the development of new services and applications. This transformation necessitates increased network capacity and throughput, as well as the integration of numerous technologies. Flawless operations and management have always been a problem for diverse wireless networks, but many service providers have met client demands. 5G is altering the cellular sector by offering new economic options, opening doors to new services, and introducing innovation to replace traditional techniques. It should be sufficiently capable and autonomous to cope with the changing demand for QoS to handle the application-driven network dynamically [1], [2].

3GPP considers network slicing as a key enabler for 5G to support new business models and user experiences. Slicing enables operators to deploy several network instances on a single infrastructure to provide heterogeneous QoS to different applications, use cases, and business services [3]. The International Telecommunication Union (ITU) classified 5G services into three major types [4]. Enhanced Mobile Broadband (eMBB), which serves high bandwidth requirements for High Definition (HD) video streaming, Virtual reality (VR) and Augmented reality (AR) technologies. Ultra-reliable and Low-latency Communications (uRLLC) focus on latency-sensitive services, such as Assisted and Automated Driving, and Remote Management. Massive Machine Type Communications (mMTC) meet the requirements for densely connected devices to realize Smart City and Smart Agriculture. To prevent QoS degradation, each user request must be assigned to a specific slice type, such as eMBB, uRLLC, or mMTC. For example, traffic coming from IoT devices need to be assigned to mMTC slice. In addition, a single device can simultaneously use multiple services, each of which may need to be assigned to distinct network slices. However, a user request may not specify which network slice type to be used to serve the request. In such a scenario, a network operator needs to classify a user request to a specific slice based on its QoS requirements and device characteristics using accurate prediction.

User-request data can be utilized in two ways, i) collect prior user requests as a dataset containing all the important features, ii) consider real-time user requests as they arrive in online manner. In this paper, we first replicated the result of [3] for a smaller dataset of 65K user requests using Deep Neural Network (DNN) for predicting network slice type. Afterwards, we used the extended dataset available in another repository[1] with more than 466K records for the same classification task. Our first contribution includes engaging multiple traditional learning (Machine Learning and Deep Learning) algorithms to predict the slice types based on incoming user requests with specific QoS requirements. We also utilized Cross Validation [5], L1 and L2 Regularization [6], and Dimension Reduction using PCA [7] as feature engineering tasks to identify the most important features. Furthermore, there are challenges in working with multi-class targets despite classification algorithms and libraries being the subject of extensive research. We modified the sklearn library such that cross-validation could be applied to the multi-class target data. We used grid search to fine-tune the hyper-parameters of the models to achieve the optimal outcome. Finally, our findings show slightly imbalanced classes in the target feature. We used the weighting method [8] to overcome this issue.

Our second contribution is to consider on-demand real-time user requests as opposed to utilizing an offline pre-collected dataset. The allocation of specific slices to real-time user requests has more significance as a network operator may need it for various types of applications and their users. Traditional learning with offline training may not be suitable for real-time user requests as data distribution may change frequently. To address this problem, we applied multiple online (e.g.,

[1] umkc/networkslicing5g dataset (2022-03-22)

incremental) learning techniques to predict the slice types based on continuous incoming user requests with specific QoS requirements. We obtained **99.33%** accuracy using an Incremental Learning [9] model that beats not only the prior result of [3] but also our traditional learning models.

The remainder of the paper is organized as follows: we review related works in Section 2. Dataset details and analysis, feature engineering, and applied models are presented in Section 3. Section 4 examines evaluation settings and outcomes by comparing various models and feature combinations. Finally, section 5 concludes the paper by discussing future research.

II. Related Work

Numerous efforts have been made to optimize and schedule network and radio resources in the most efficient manner. However, precise network slice allocation depending on the service or application requirements is a prerequisite for successful deployment of 5G network slices. With the combination of edge computing, software-defined network (SDN), and network function virtualization (NFV), the proposed solution in [10] creates on-demand and distributed network functions and allocates optimal load. A unique queuing strategy-oriented solution using NFV and SDN has been proposed in [11], which includes the allocation of data rates dynamically on high demand, where each slice contains numerous service levels. The "SPArTaCuS" framework [12] prioritizes network traffic using an SDN approach to adapt the situation for smart cities. It addresses the problem of network congestion in crowded places during large events and disasters with the concept of network slicing. The authors in [13] focus on issues for network slicing architecture, including network slice selection.

With the continuous advancement of Machine Learning and Deep Learning algorithms, several prediction based approaches for associating appropriate network slice with desired service to maintain the QoS have been developed. The authors in [14] use machine learning algorithms to address the slice allocation problem that finds the best network slice for giving the specific service and maintaining the QoS. Another framework for the operation and control of network slices is based on machine learning that continuously monitors workloads, throughput, and resource utilization [15]. It then adjusts the resources allocated to network slices dynamically. In this work [16], a model named GS-DHOA has been proposed to optimize the weighing function of networks. After applying GS-DHOA, the authors in [16] propose to classify network slices extracted from their dataset using deep neural networks. Another recent work has used Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM), where CNN classifies the network slice, and LSTM defines the statistics [17]. The approach, named "DeepSlice," applies deep learning neural network for network slice classification and works for unknown device types based on the availability and efficiency of the network [3]. In contrast to the state-of-the-art, in this paper we not only apply a suite of ML and deep learning classifiers combined with feature engineering for offline training but also investigate incremental learning approaches to facilitate online training with real-time user requests.

III. Methodology

A. Dataset

Different network and device-related KPIs are included in the umkc/networkslicing5g dataset (2022-03-22), including device type (IoT device, smartphone, URLLC device, and so on), category of User Equipment (UE), QoS Class Identifier (QCI), packet delay budget (latency), packet loss rate (reliability), date and time of the week among other features [3]. The network Slice Type is used as a target variable. This data has been captured by sending control packets between the UE and the network. There are more than 466k records in the dataset. The total count for each class reflects a slight imbalance among the target classes (Network Slice Types) in the dataset (The record counts of uRLLC, mMTC, and emBB classes are 209K, 131K and 125K, respectively).

B. Data Analysis & Feature Engineering

The dataset consists of approximately 466,000 records with eight input features and one target feature. Table I displays the features and their respective details.

Also, a correlation analysis of the data was performed. Fig.1 shows that the dataset is moderately correlated according to the value of Pearson Correlation Coefficient [18]. Based on this observation, we applied dimensionality reduction [19] to achieve better results.

Fig. 1: Feature Correlation Analysis

For feature analysis, RF [20] with hyperparameter tuning using grid search was used. Fig.2 depicts the feature importance graph. As indicated by Fig.2, the Technology supported and Packet Loss Rate are the most important features, while the day, time, and QCI have lower importance.

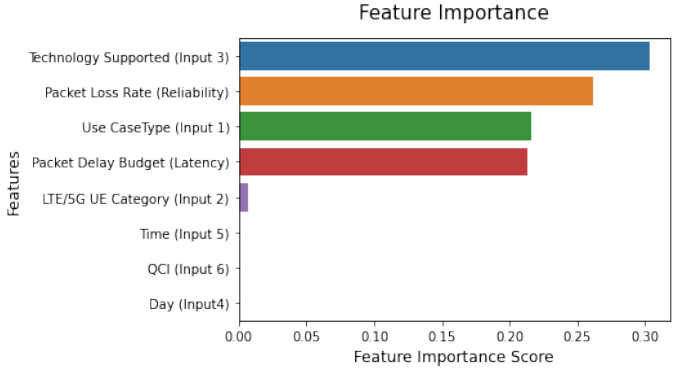


Fig. 2: Feature Importance Analysis

TABLE I: Dataset Features with Details

Features	Data type	Unique values	Modeling Parameters
Use Case Type (Input 1)	object	8	Devices (Smartphone, IoT Devices, Smart Transportation, Industry 4.0, AR/VR/Gaming, Healthcare, Public Safety/E911 and Smart City/Home)
LTE/5G UE Category (Input 2)	object	23	Category type 0-20, M1 and NB-IoT
Technology Supported (Input 3)	object	2	LTE/5G and IoT(LTE-M, NB-IoT)
Day (Input 4)	object	7	Monday-Sunday
Time (Input 5)	integer	23	23 hours
QCI (Input 6)	integer	13	Values from 1-9, 65, 66, 69, 70
Packet Loss Rate (Reliability)	float	23	0.01, 0.001, 0.000001
Packet Delay Budget (Latency)	object	3	<10ms, <50ms, <300ms
Slice Type (Output)	object	3	eMBB, mMTC, URLLC

C. ML and Deep Learning Algorithms

The Deep Learning Model (CNN) [21] was used as the baseline, inspired by [3]. The assignment of incoming user requests to three network slice classes constitutes a multi-class classification. We applied several ML Models, including RF [20], DT [22], and LR [23] with hyperparameter tuning. Since the dataset may be susceptible to overfitting, we used Cross Validation, Regularization, and Dimensionality Reduction to avoid it. For the multi-class classification, we conducted modifications to the sklearn library so that cross-validation could be applied to the multi-class target. For Incremental Learning, Multilayer Perception Classifier (MLP) [24], Stochastic Gradient Descent (SGD) [25], Hoeffding Tree Classifier [26], and Passive Aggressive Classifier [27] were used.

IV. Results and Discussion

A. Environment Settings

The dataset was divided into a training set, a test set, and a validation set containing 70%, 20%, and 10% of the data for traditional learning, and 90%-to-10% for learning and validation for incremental learning. In incremental learning, 10% of the data was retained in order to validate the learning performance outcomes and establish a proper inference time. The incremental learning process was conducted sample-by-sample and batch-by-batch (7 batches). In the sample-by-sample approach, learning was conducted one sample at a time and then evaluated with a validation set. We evaluated the accuracy of the ML models by applying them to each dataset five times and presenting the average results. All experiments are conducted on a computer with an 11th-generation Intel Core i7 processor, one GeForce RTX 3090 GPU with 24GB of RAM, and 32GB of main memory.

B. Compared Method & Performamce Metrics

We applied the Deep Learning (CNN) [20] and ML models integrated with five different combinations of input features. The resulting variants evaluated in this paper are shown in TABLE II. Specifically, we applied all input features (denoted as F), the top five crucial features discussed in section C (represented as FI5), all features except latency (represented as FWL), all features except reliability (denoted as FWR), and all features except latency and reliability (represented as FWLR) to CNN, RF, DT, and LR. The incremental learning was conducted in a sample-by-sample manner, and the best outcome was compared to the batch-by-batch method. All input features were used for incremental learning due to their superior performance in traditional learning, as demonstrated in the next section. With the following equations, Accuracy, Precision, Recall, and F1 Score were used as the performance metrics [2]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

Where TP, TN, FP, and FN stand for True Positive, True Negative, False Positive, and False Negative, respectively.
Given that the class distribution within the dataset is not balanced, weighted metrics were utilized. These metrics involve calculating the average performance metric for each class and weighting it by the number of samples present in that class.

TABLE II: Combination of Different Traditional Learning Classifiers and Features

	F	FI5	FWL	FWR	FWLR
CNN	CNN+F	CNN+FI5	CNN+FWL	CNN+FWR	CNN+FWLR
RF	RF+F	RF+FI5	RF+FWL	RF+FWR	RF+FWLR
DT	DT+F	DT+FI5	DT+FWL	DT+FWR	DT+FWLR
LR	LR+F	LR+FI5	LR+FWL	LR+FWR	LR+FWLR

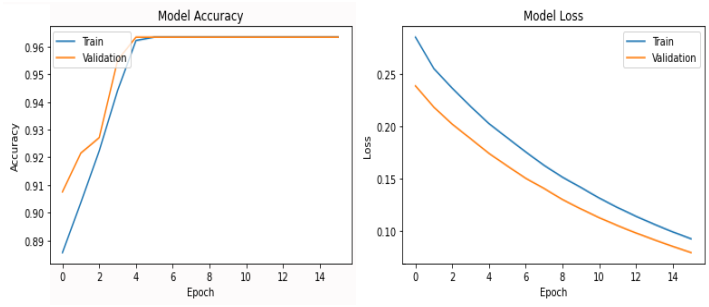
C. Results and Analysis

As mentioned in Section I, we applied the DL model on the 65K network slicing data and got 95.39% accuracy over their 95%. Then we used RF, DT, LR, and CNN models on the extended dataset with more than 466K rows. TABLE III shows the accuracy, precision, recall, and F1 score for all the combinations of features and models stated in TABLE II. **DT+F** outperformed all scenarios with 98.65% accuracy. The table shows that performance of all four models in combination with five important features (FI5) is very close to those for models with all features, suggesting to use of the most important features. When we removed important features such as latency, reliability, or both, the performance significantly dropped. We also report training and inference time in TABLE III to see if the training could be completed online or offline. After applying all of the combinations of ML and DL algorithms, we observed that training should be done offline because training takes few minutes. However, inference time is significantly shorter, allowing the trained model to use in real-time without compromising performance.

Cross-validation, regularization, and dimensionality reduction were used to overcome the overfitting issue. The graphs for

TABLE III: Traditional Learning Results

M+F	Accuracy	Precision	Recall	F1 Score	Training Time (s)	Inference Time (ms)
CNN+F	96.12	95.88	96.74	96.04	250	0.32
RF+F	97.87	**98.85**	97.85	97.84	53	0.26
DT+F	**98.65**	98.79	98.41	**98.57**	49	0.25
LR+F	94.26	95.56	94.44	94.03	26	**0.22**
CNN+FI5	96.1	95.89	96.73	96.03	218	0.32
RF+FI5	97.86	98.84	97.86	97.85	39	0.26
DT+FI5	98.62	98.78	**98.42**	98.56	47	0.25
LR+FI5	94.14	95.48	94.39	93.92	**23**	**0.22**
CNN+FWL	80.11	83	76.27	80.43	261	0.32
RF+FWL	69.64	76.02	70.9	68.38	62	0.26
DT+FWL	74.28	80.6	75.55	72.8	57	0.25
LR+FWL	63.98	74.31	65.71	59.93	39	0.22
CNN+FWR	91.46	91.46	91.44	91.67	258	0.32
RF+FWR	86.67	88.66	87.07	86.35	57	0.26
DT+FWR	88.89	90.74	89.26	88.59	54	0.25
LR+FWR	74.62	71.93	76.47	70.98	24	**0.22**
CNN+FWLP	72.95	72.96	72.94	73.02	274	0.32
RF+FWLP	51.5	44.51	52.45	46.09	107	0.27
DT+FWLP	52.63	53.4	54.44	47.54	82	0.25
LR+FWLP	61.32	64.2	61.89	57.97	30	**0.22**



(a) Model Accuracy (b) Model Loss

Fig. 3: CNN Model Accuracy and Loss per Epoch

accuracy and loss over training and validation for the CNN model shown in Fig.3a and Fig.3b demonstrate that the model does not exhibit overfitting when utilizing the mentioned techniques. Due to its superior performance, DT+F was selected as the model and feature combination for further analysis. We report the class-based accuracy of this combination in TABLE IV. As shown in TABLE IV, mMTC performed the best among the three network slice types with **99.42%** accuracy. This is due to the imbalance class distribution as shown in TABLE IV that can be addressed in a future work.

TABLE IV: Class Accuracy

Class	Total Count	Test Set Count	Percentage	DT Accuracy
uRLLC	209300	41944	0.45	98.57
mMTC	131859	26533	0.28	**99.42**
eMBB	125580	24871	0.27	97.98

Relying on the above observation, we applied incremental learning algorithms due to their compatibility with continuous user incoming requests. TABLE V contains the accuracy, precision, recall, F1 score, training Time, and inference Time for the MLP, SGD, HT, and PA classifiers. Among all the classifiers, **SGD** performed the best with the accuracy of **99.33%**. Incremental learning training time is slightly longer than traditional learning since training and evaluation are performed in each iteration. In general, Incremental learning outperforms traditional learning in terms of other performance metrics. This is due to its dynamic computation, which evolves over a large number of samples.

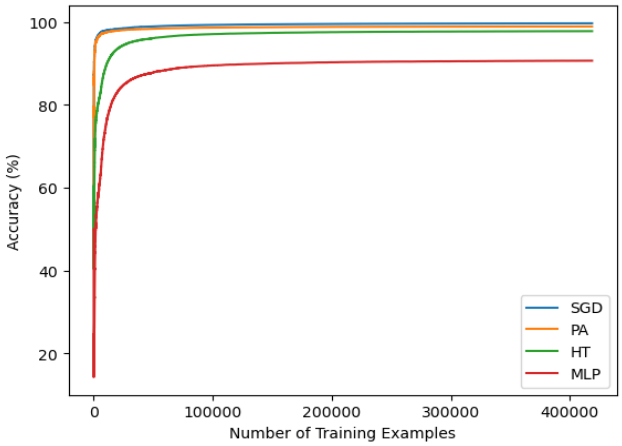


Fig. 4: Incremental Learning Classifiers Comparison

Fig. 4 shows that the classifiers' performance might be enhanced with further training instances, despite their slope becoming quite modest after certain points. Due to its reasonable inference time and highest accuracy, the SGD might be considered the best-performing online learning classifier applicable to real-time scenarios. We also compared incremental learning on a sample-by-sample and batch-by-batch basis. For the comparison, the SGD classifier was chosen due to its superior performance. According to TABLE VI, although the batch-by-batch approach has a shorter training time, its performance is slightly worse than the sample-by-sample case in terms of other performance metrics.

TABLE V: Incremental Learning Sample by Sample

Classifier	Accuracy	Precision	Recall	F1 Score	Training Time (s)	Inference Time (ms)
MLP	91.15	92.62	91.16	90.74	94	0.1
SGD	**99.33**	**99.32**	**99.33**	**99.33**	120	**0.05**
HT	99.12	99.14	99.11	99.12	**104**	0.06
PA	99.16	99.02	99.15	99.16	106	0.07

TABLE VI: Incremental Learning Batch by Batch

Classifier	Accuracy	Precision	Recall	F1 Score	Training Time (s)	Inference Time (ms)
SGD	98.93	98.88	98.89	98.92	62	0.05

V. Conclusion

In this paper, we address the problem of classifying incoming user requests with specific QoS requirements to the most suitable network slice types. The optimal mixture of traditional learning classifiers and feature sets was evaluated and compared to the incremental learning method. Using the SGD classifier for incremental learning and all the input features, the highest accuracy (**99.33%**) with the best inference time (**0.05 ms**) was achieved, enabling its implementation in real-time streaming data. Future research directions will involve extending this work with additional QoS-related forecasts, such as predicting future network traffic or load, and handovers.

REFERENCES

[1] Aljiznawi, R. A., Alkhazaali, N. H., Jabbar, S. Q., and Kadhim, D. J. (2017). Quality of service (QoS) for 5g networks. International Journal of Future Computer and Communication, 6(1), 27.

[2] Grandini, M., Bagli, E., and Visani, G. (2020). Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756.

[3] Thantharate, A., Paropkari, R., Walunj, V., and Beard, C. (2019, October). DeepSlice: A deep learning approach towards an efficient and reliable network slicing in 5G networks. In 2019 IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON) (pp. 0762-0767). IEEE.

[4] Huawei. (2016). 5G Network Architecture—A High-Level Perspective.

[5] Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., and Ridella, S. (2012). The 'K'in K-fold cross validation. In 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) (pp. 441-446). i6doc. com publ.

[6] Moore, R. C., and DeNero, J. (2011). L1 and L2 regularization for multiclass hinge loss models.

[7] Zhang, T., and Yang, B. (2016, November). Big data dimension reduction using PCA. In 2016 IEEE international conference on smart cloud (SmartCloud) (pp. 152-157). IEEE.

[8] Willems, F. M., Shtarkov, Y. M., and Tjalkens, T. J. (1995). The context-tree weighting method: Basic properties. IEEE transactions on information theory, 41(3), 653-664.

[9] Gepperth, A., and Hammer, B. (2016). Incremental learning algorithms and applications. In European symposium on artificial neural networks (ESANN).

[10] Ma, L., Wen, X., Wang, L., Lu, Z., and Knopp, R. (2018). An SDN/NFV based framework for management and deployment of service based 5G core network. China Communications, 15(10), 86-98.

[11] Kurtz, F., Bektas, C., Dorsch, N., and Wietfeld, C. (2018, June). Network slicing for critical communications in shared 5G infrastructures-an empirical evaluation. In 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft) (pp. 393-399). IEEE.

[12] Abhishek, R., Zhao, S., and Medhi, D. (2016, September). Spartacus: Service priority adaptiveness for emergency traffic in smart cities using software-defined networking. In 2016 IEEE international smart cities conference (ISC2) (pp. 1-4). IEEE.

[13] Yoo, T. (2016, October). Network slicing architecture for 5G network. In 2016 International Conference on Information and Communication Technology Convergence (ICTC) (pp. 1010-1014). IEEE.

[14] Gupta, R. K., and Misra, R. (2019, December). Machine learning-based slice allocation algorithms in 5G networks. In 2019 International Conference on Advances in Computing, Communication and Control (ICAC3) (pp. 1-4). IEEE.

[15] Kafle, V. P., Martinez-Julia, P., and Miyazawa, T. (2019). Automation of 5G network slice control functions with machine learning. IEEE Communications Standards Magazine, 3(3), 54-62.

[16] Abidi, M. H., Alkhalefah, H., Moiduddin, K., Alazab, M., Mohammed, M. K., Ameen, W., and Gadekallu, T. R. (2021). Optimal 5G network slicing using machine learning and deep learning concepts. Computer Standards and Interfaces, 76, 103518.

[17] Khan, S., Khan, S., Ali, Y., Khalid, M., Ullah, Z., and Mumtaz, S. (2022). Highly Accurate and Reliable Wireless Network Slicing in 5th Generation Networks: A Hybrid Deep Learning Approach. Journal of Network and Systems Management, 30(2), 1-22.

[18] Schober, P., Boer, C., and Schwarte, L. A. (2018). Correlation coefficients: appropriate use and interpretation. Anesthesia and Analgesia, 126(5), 1763-1768.

[19] Sorzano, C. O. S., Vargas, J., and Montano, A. P. (2014). A survey of dimensionality reduction techniques. arXiv preprint arXiv:1403.2877.

[20] Rogers, J., and Gunn, S. (2005, February). Identifying feature relevance using a random forest. In International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection" (pp. 173-184). Springer, Berlin, Heidelberg.

[21] Sekaran, K., Chandana, P., Krishna, N. M., and Kadry, S. (2020). Deep learning convolutional neural network (CNN) With Gaussian mixture model for predicting pancreatic cancer. Multimedia Tools and Applications, 79(15), 10233-10247.

[22] Safavian, S. R., and Landgrebe, D. (1991). A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics, 21(3), 660-674.

[23] Tsangaratos, P., and Ilia, I. (2016). Comparison of a logistic regression and Naïve Bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size. Catena, 145, 164-179.

[24] Kubat, M. (1999). Neural networks: a comprehensive foundation by Simon Haykin, Macmillan, 1994, ISBN 0-02-352781-7. The Knowledge Engineering Review, 13(4), 409-412.

[25] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[26] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In KDD'01, pages 97–106, San Francisco, CA, 2001. ACM Press.

[27] Crammer, Koby and Dekel, Ofer and Keshet, Joseph and Shalev-Shwartz, Shai and Singer, Yoram. (2006). Online Passive-Aggressive Algorithms. Journal of Machine Learning Research. 7. 551-585.